User’s Guide

TREND 3.6

DeskTalk

DeskTalk Systems, Inc. makes no representation or warranties with respect to this manual, or except as specifically stated in the applicable user agreement or warranty notice, with respect to any hardware, firmware, or software described in this manual. DeskTalk Systems, Inc. specifically disclaims any expressed or implied warranties or merchantability, title, or fitness for a particular purpose. Furthermore, DeskTalk Systems, Inc. reserves the right to make revisions or changes to any and all parts of the manual, hardware, firmware, or software at any time without obligation to notify any person or entity of the changes.

TREND 3.6 User’s Guide
Publication No. UG361-2000JUN16

DeskTalk Systems, Inc.
19191 South Vermont Avenue
Ninth Floor
Torrance, CA 90502
(310) 630-1000

Contents

TREND

Figures

TREND

Preface

This manual is intended for anyone using TREND network management software. It describes the philosophy of operation and hands-on use of the product. It also helps you to interpret the data provided and to maintain TREND.

Document Overview

This user’s guide is designed to provide both tutorial instructions and reference information. It has 22 chapters and 6 appendixes:

Chapter 1 Introduction to TREND

Provides a general description of TREND functionality, including the background information needed to understand how TREND works.

Chapter 2 TREND Configuration for trendadm

Shows the administrative user *trendadm* how to define users and groups and create (polling) views made up of different groups, nodes, and types for data collection; and how to change defaults or rollup and aging parameters for data in the database. trendadm can also truncate and delete tables in the database. This chapter also discusses the issues related to maintaining the TREND license key.

Chapter 3 The TREND Main Window

Describes the main window for launching TREND application modules and defining network configuration data.

Chapter 4 Autopilot

Describes TREND's automated approach to report generation, which uses ReportPacks. See the TREND ReportPack Guide for the current version of this chapter.

Chapter 5 Discover

Describes the Discovery module, which automates the discovery of network nodes.

Chapter 6 MIBwalker

Describes TREND's MIBwalker module, which you use to load and view MIBs and define SNMP data collection.

Chapter 7 Collect Data

Describes the Collect Data module, which you use to define polling policy for data collection.

Chapter 8 Data Manager

Describes how to use Data Manager's tabular display to administer and monitor TREND tables.

Chapter 9 Data Aggregation and Manipulation

Provides an overview of TREND data aggregation and manipulation processes and introduces the utilities that perform these functions: TRENDit, TRENDstep, TRENDsum, and TRENDrank. Relates data aggregation concepts to the TREND ReportPacks.

Chapter 10 TRENDbuild

Describes how to create and manipulate query files, which the TRENDgraph, TRENDsheet, and Grade of Service applications use.

Chapter 11 TRENDsheet

Describes how to use TRENDsheet to create, manipulate, and print tabular data displays and export their data to an ASCII file.

Chapter 12 TRENDgraph

Describes how to use TRENDgraph's point-and-click graphical user interface to create custom graphical reports.

Chapter 13 Grade of Service Reporting

Describes how to use the GOS application to create reports on system health in a stacked bar format.

Chapter 14 Report Launcher

Describes how to use the Report Launcher to run reports on an ad hoc basis.

Chapter 15 Report Scheduler

Describes how to use the Report Scheduler to schedule periodic printing of reports.

Chapter 16 RMON Support

Describes the setup and polling policy definition for RMON probes including support for RMON2 extensions.

Chapter 17 RMON2 Support

Describes the RMON2 DataPipe.

Chapter 18 TRENDcopy

Describes TRENDcopy, which allows you to copy information from one database to another.

Chapter 19 TRENDlabel

Describes the command-line driven TRENDlabel application, which you can use to populate specific columns of key tables.

Chapter 20 TRENDexporter

Describes the TREND exporting module, which adds versatility to data export.

Chapter 21 TRENDproc

Describes TRENDproc, which allows you to streamline the reporting process.

Chapter 22 TRENDdbview

Describes TRENDdbview, which generates SQL database views.

Appendix A Man Pages

Contains the latest Man Pages published by DeskTalk Systems, Inc.

Appendix B Database Administration

Provides the TREND Administrator (trendadm) with useful information for maintaining the TREND database.

Appendix C Performance Tuning Issues

Discusses how to improve TREND performance.

Appendix D Setting Up a Satellite Server Environment

Discusses considerations for processing performed on the central server and satellite servers.

Appendix E Discovery Considerations

Describes how TREND stores nodes and how you can modify Discovery files.

Appendix F Relocating the TREND Database

Explains how to move the TREND database from one host to another.

Related Documents

The following documents may also be useful:

TREND Manuals

TREND Installation Guide
TREND User's Guide
TREND ReportPack Guide
TRENDweb Installation and Administration Guide
Guide To Building TREND Reports
Guide To Viewing TREND Reports

Sybase Manuals

SQL Server Transact - SQL User's Guide
Sybase SQL Server System Administration Guide
SQL Server Reference Manual: Volume 1, Commands, Functions, and Topics
SQL Server Reference Manual: Volume 2, System Procedures and Catalog Stored Procedures
SQL Server Troubleshooting Guide

Standards Documents

Structure and Identification of Management Information for TCP/IP-Based Internets, RFC 1155
Simple Network Management Protocol, RFC 1157
Concise MIB Definitions, RFC 1212
Management Information Base for Network Management of TCP/IP-Based Internets

Additional Publications

Case, J., M. Fedor, M. Schoffstall, and J. Davin, *A Simple Network Management Protocol*, RFC 1098, University of Tennessee at Knoxville, NYSERNET, Rensselaer Polytechnic Institute, Proteon, August 1988

Rose, M.T., *The Simple Book: An Introduction to Internet Management*, Second Edition, PTR Prentice Hall, 1994

Stallings, W. SNMP, SNMPv2, and CMIP: *The Practical Guide to Network Management Standards*, Addison Wesley, October 1993

Waldbusser, S., *Remote Network Monitoring Management Information Base*, RFC 1271, Carnegie Mellon University, November 1991

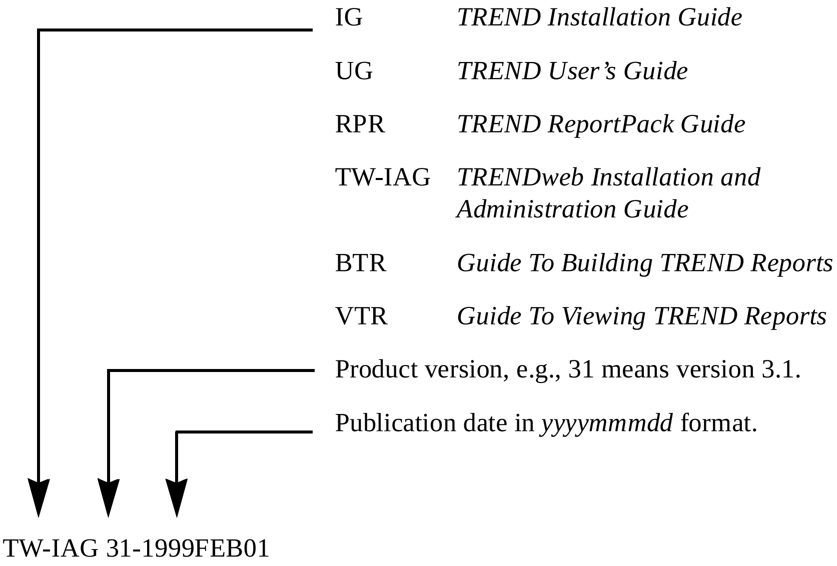
Bowman, J., Emerson, S., Darnovsky, M., *The Practical SQL Handbook,* Second Edition, Addison-Wesley, 1993

Online User Manual Availability

TREND is shipped with online-viewable user manuals in Portable Document Format (PDF), along with the Adobe® Acrobat® Reader. These documents and the Adobe Acrobat Reader are available prior to installation in the Documentation directory of the installation CD. You need to install the Adobe Acrobat Reader to view these documents online.

These documents are also installed onto your hard drive in the docs directory in DPIPE_HOME during installation.

You can download the latest TREND manual updates from the DeskTalk Systems Customer FTP site. Look at the document number (on the manual's title page) to determine if you have the latest version of manual:



IG	*TREND Installation Guide*
UG	*TREND User's Guide*
RPR	*TREND ReportPack Guide*
TW-IAG	*TRENDweb Installation and Administration Guide*
BTR	*Guide To Building TREND Reports*
VTR	*Guide To Viewing TREND Reports*

Product version, e.g., 31 means version 3.1.

Publication date in *yyyymmmdd* format.

TW-IAG 31-1999FEB01

Call DeskTalk Systems Customer Support for download instructions.

TREND

1 Introduction to TREND

Desktalk System’s TREND is a network management software product that collects network information from SNMP agents, stores it in a commercial relational database or multiple databases, manages data, and provides historical reporting based on that information.

TREND is comprised of a set of generic application modules for data collection, aggregation, management and reporting, as well as specific SNMP and RMON modules. TREND is a complete application that any member of the network staff can use to collect appropriate data and produce meaningful reports, without networking or database expertise. Its extensible architecture also provides an application toolkit that the network expert can use to extend TREND’s functionality without programming or SQL scripting.

TREND is a true client/server, scalable SNMP application for enterprise network management. TREND’s report-building capability allows system operators and network managers to create custom reports and report libraries. The RDBMS data repository can be centralized or distributed across multiple databases, while the report builder and table administration tool eliminate the need for RDBMS expertise.

TREND Overview

As shown in Figure 1-1: TREND Applications and Processes, TREND is comprised of user-visible application modules and user-transparent processes that carry out the required tasks.

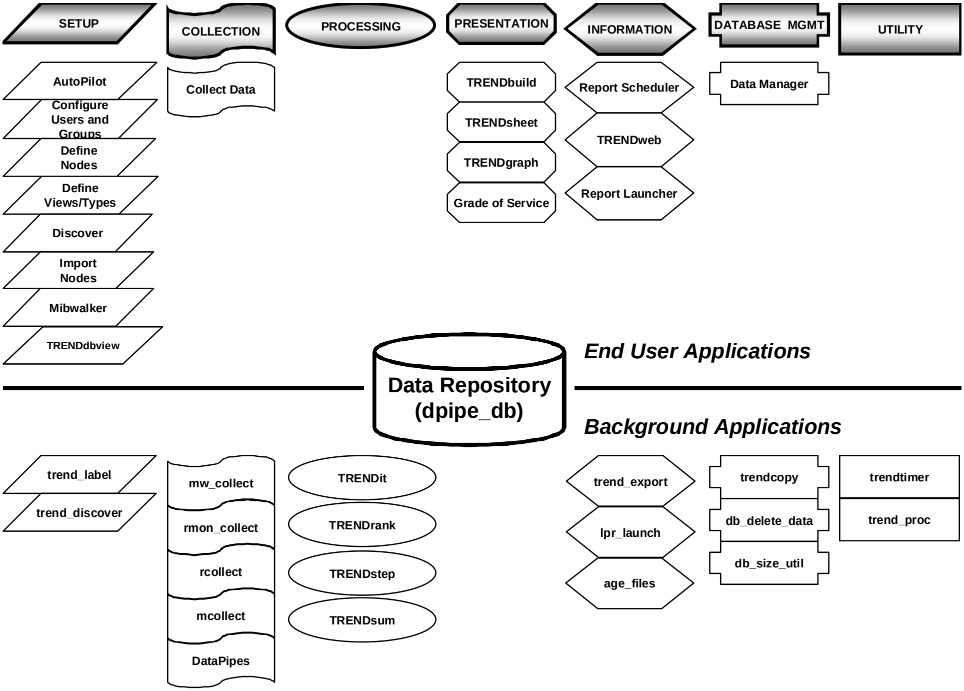


Figure 1-1: TREND Applications and Processes

TREND Layer	**GUI Applications**		**Background Processes**	
Information	Report Schedule	->print.N	lpr_launch	<-print.N
				->.ps/.gif/.bmp
			trend_export	<-cmd file
				->data file
	Report Launcher			
	TRENDweb			
			age_files	
Presentation	TREND Build	->.qss/.qgr/.gos		
	TRENDsheet	<->.qss files		
	TRENDgraph	<->.qgr files		
	Grade of Service	<->.gos files		
Processing			TRENDit	
			TREND_sum	<-.sum file
			TRENDstep	<-.rot
			TRENDrank	<-.rnk
Collection	Collect Data		DataPipes	
			ee_collect	ecollect
			mw_collect	mcollect
			rmon_collect	rcollect
Setup	Configure Users/Groups			
	Discover		trend_discover	
	Import Nodes	<-node list file	trend_lbel	
	Autopilot	<-package file		
	Define Nodes			
	Define Views/Types			
	MIBwalker	.mib file		
	TRENDdbview	->.vwb files		

TREND Layer	GUI Applications		Background Processes	
Database Mgmt	Data Manager		trendcopy db_delete_data db_size_util	
Utilities			trendtimer trend_proc	<-trendtimer.sched <-proc file

This release of TREND includes the following modules:

Module	Description
Collect Data	An application for defining, reviewing, and updating an enterprise polling policy for periodic collection of management data by TREND polling agents.
Data Manager	Data Manager is an administrative tool that displays the structure and use of the TREND database at a glance. Tables can be truncated or deleted, and default or table specific aggregation and aging choices can be set through the GUI. Data Manager also displays the percentage of database space being used.
db_delete_data	TREND's data aging process is run on a periodic basis. Data older than the retention time specified are deleted on a day boundary each time the db_delete_data process runs.

(1 of 5)

Module	Description
Grade of Service	The Grade of Service (GOS) application converts historical performance statistics into meaningful graphical displays using a weighted, stacked bar chart format. GOS allows the user to select any set of SNMP or RMON metrics and assign a grading scale to each metric based on relative importance. GOS can be used for reporting on routers, servers, LAN segments, WAN links, or other network components where system health is critical.
MIBwalker	MIBwalker extends the polling capabilities of TREND, allowing the advanced TREND user to select new groups of MIB variables and add them to the database schema. MIBwalker also provides conventional MIB browsing via its MIB tree structure and point-and-click interface.
mw_collect	TREND’s primary polling agent, for collection of SNMP data from nodes on your network.
Report Launcher	Report Launcher is used to display reports by selecting a query file and related network element for which data has been collected.
Report Scheduler	Report Scheduler is used to set up groups of pre-defined reports to be printed for groups of devices on a daily or weekly basis.
rmon_collect	rmon_collect is a specialized polling agent tailored to the unique RMON MIB. It collects RMON history, host, matrix, and statistics groups and supports RMON2 network and application layer functionality via AXON ECAM, Netscout Systems Domain View, and other extensions.

(2 of 5)

Module	Description
TRENDbuild	TRENDbuild lets the user create meaningful graphs and tabular reports without using SQL. Using a point-and-click GUI interface to select variables, assign custom aliases and build expressions, TRENDbuild generates report definitions, which are used by TRENDgraph, TRENDsheet, or Grade of Service. These report definitions are query files that can be stored anywhere the user chooses, allowing creation of report libraries.
trendcopy	TRENDcopy provides incremental, periodic, filtered database replication to a backup server without the assistance of an experienced database administrator. Replication can be ad-hoc using a command line interface or scheduled on a periodic basis. Database tables can be created and/or incrementally copied to another database based on user-definable criteria such as thresholds, time spans or sets of nodes, so that the database contains only the information required for reporting applications.
TRENDdbview	A GUI application (though launched from the command line) that enables TREND users to create database views at will.
trendexport	A command line application that allows TREND users to export TREND data as ASCII text. This makes it possible to use the data generated by TRENDit and TRENDsum in disparate applications, such as spreadsheets.

(3 of 5)

Module	Description
TRENDgraph	TRENDgraph reports provide time-based line graphs of information selected from the data repository based on queries defined through TRENDbuild. The user can display multiple values from a single device or multiple devices (or interfaces), all synchronized by time for easy comparison. The user can select line and data point color, thickness and pattern, X-axis time interval, Y-axis data range and labels, and graph headings. Cumulative area and stacked bar formats are also supported. TRENDgraph will print the displayed graph to a postscript printer or a file.
TRENDit	Meaningful information is produced by aggregating raw data into periodic summaries. The TRENDit process creates hourly, daily, weekly, and monthly rollup tables from polled data. Each summary provides average, maximum, minimum and total values for counter and gauge fields and carries forward character data and constants.
trendlabel	A command line application that allows TREND users to label various column headers.
trendproc	A simple language for process chaining. TRENDproc makes it possible to run various TREND processes sequentially, in parallel, or both.
TRENDrank	A command line application that builds a ranked table on selected metrics from a summary table. TRENDrank enables you to rank and compare selected metrics from the current period with the appropriate day of the week from the baseline summary.

(4 of 5)

Module	**Description**
TRENDsheet	TRENDsheet reports display rows and columns of information selected from the data repository or calculated from stored values. The data can be sorted on any combination of fields, and columns can be resized and reordered directly on the screen. Printouts of reports can be requested, or the data can be exported to an ASCII format file. Report definition parameters are stored in a file that can be stored anywhere the user chooses, allowing creation of a report library.
TRENDstep	A command line application that conditionally processes each row of a rate input table and stores the results in a new output table. This processing enables you to combine data from multiple rows of an input table into a single row in the output table and compute new columns of data in the output table.
TRENDsum	An application that takes the cleaned rate data provided by TRENDit and summarizes it into many useful statistics.
trendtimer	A process that controls the collection, trending and deleting of data. The trendtimer.sched file contains all the entries for mw_collect, rmon_collect, ee_collect, TRENDit, db_delete_data, and lpr_launch.

(5 of 5)

A Brief Tour of TREND

Collecting Data

The Collect Data application is used to define enterprise data polling policy. It was designed specifically to handle SNMP data structures (MIBs). A polling policy is a set of requests to collect groups of MIB variables from one or more devices at regular intervals. The actual polling of data is carried out by TREND's polling agents: ee_collect, mw_collect and rmon_collect. Launched periodically, these agents determine their polling activities based on the defined polling policy.

TREND users can collect specific MIB data from any SNMP or RMON agent. Pollers collect data and deposit the data into the server database. The server contains the data repository with polling information, data tables, and indexed key tables.

Note: TREND servers can be physically distributed across the enterprise. It is also possible to configure multiple servers, with a server-poller combination (see Satellite Database Servers below).

TREND employs specialized modules for defining and implementing an SNMP polling policy. Collected data is bulk copied into a designated database using a common schema. The common database schema allows data from multiple sources, in multiple formats, to be processed and used for reporting by TREND.

Using the MIBwalker tool, the user can browse MIB structures and define additional data polling requests on an ad hoc basis. The particular group of MIB variables, or even the entire MIB group selected will be added to the TREND database for future polling requests. MIBwalker uses the Collect Data application and the mw_collect process to enact data polling.

Once collected, raw data must be reduced into a more usable form. Data reduction in TREND has been designed to meet several goals, including speeding reporting, producing a schema in which simple queries produce powerful results, and maintaining a manageable database size. TREND reduces data by rolling it up over time, or aggregating it. The TRENDit process performs data aggregation in TREND. Data aggregation is performed using a series of steps, typically leading to longer retention of more highly aggregated data.

Raw data tables in TREND are populated with timestamp values, constants and statistics. Statistics may be counters, integers, strings, or gauge values. A counter is a continuous raw count of a statistic since a device was rebooted. A gauge is an instantaneous measurement that is only meaningful at an instant in time.

Rate tables are created by selecting data from raw data tables and calculating delta values between successive points. Periodic aggregations rolls-up average, total, minimum and maximum values by the hour, day, week, and month. Obviously, data tables cannot be created until sufficient time has been allowed for data to be collected -- aggregated data (hourly, daily, weekly, and monthly) will not be available until after TREND’s daily aggregation process runs. In general, you must wait until at least the next day after beginning data collection to start producing reports.

Reporting

TREND reporting is based on the creation and manipulation of query files. These files contain all of the information necessary to define, display and print tabular and graphical reports. TREND reports can be created using the TRENDbuild application, which is accessed by both the grapher (TRENDgraph), spreadsheet (TRENDsheet), and Grade of Service applications. For TRENDsheet, the query file defines what columns will be displayed, their order and name, and expressions used to combine column values. Similarly, TRENDgraph files contain line and point color, pattern thickness, as well as other attributes.

The TREND report (query) files are handled like any other files (e.g., word processing files or spreadsheets) on your system. For example, a file could be stored in a standard default directory or subdirectory linked to a central library of query files. A number of report types and formats are available.

Graphical Reports

By executing the TRENDgraph module from the main menu, you can browse and select from a list of pre-formatted reports, called query files. Query files contain the SQL commands, which retrieve data from specific tables in the database. Once a query file has been specified, the user can select a variety of options to specify the node, group, or instance; the date and day range; X and Y axis scaling; statistic selection; graphic display attributes, and graph rendering (i.e., line, stacked bar, or area graph). Graphical reports can be printed directly to postscript printers, scheduled for periodic execution, or saved to files for customized post-processing, such as conversion to Web page format. Drill-down reporting allows TRENDgraph users to point at an area in any TREND graph and click to produce a detailed tabular report. The TRENDsheet display is filtered to include the objects and instances associated with the selected bar graph. This feature is particularly useful in the grade of service application. Instances of performance that are below a certain level can be analyzed in detail by drilling down.

Tabular Reports

By executing the TRENDsheet module from the main menu, you can browse and select from a list of pre-formatted spreadsheet-style reports. Like graphical reports, TRENDsheet reports are based on query files. Tabular display of selected statistics can be sorted and displayed with a number of options. Output can be printed to postscript printers or saved to a file, like TRENDgraph. TRENDsheet also supports exception reporting. Exception reporting allows you to define constraints, using Boolean expressions, as conditions for executing a particular tabular report.

Grade of Service (GOS) Reports

GOS reporting is a valuable tool for network and systems health reporting and Service Level Agreements (SLAs). The GOS algorithm converts historical performance statistics into instantly meaningful graphical displays using a weighted stacked bar chart format. GOS allows the user to select any set of SNMP or RMON metrics and assign a grading scale to each selected metric based on relative importance. An overall grading scale (e.g., excellent, good, fair, or poor) is applied to the weighted metrics. This information allows network service providers to concentrate on areas where performance is substandard. Since GOS can be set up with metrics from any SNMP MIB, users can develop customized GOS reporting for routers, servers, LAN segments and WAN links.

Custom Reports

The Report Scheduler module, selectable from the main menu, allows you to create customized charts and graphs using a point-and-click interface. You select the data source and the set of statistics you want in your report. TRENDbuild allows you to assign aliases to MIB statistic names, and build computed variables using selected statistics, constants, and arithmetic operators. Expressions are very useful for computing relative utilization, for example. Once you have created the report specifications, you can specify the name of the report query file.

Report Launcher

The Report Launcher is an easy way to run an existing query file against a network element. This interface allows you to select a report group (e.g., routers), report type (e.g., GOS), and a network element.

Report Chaining

Related reports can be linked together for automatic printing via the Report Scheduler option. A series of related “drill-down” reports provide a detailed picture that is unavailable from any single report.

TRENDweb Report Building and Viewing Facilities

TRENDweb 3.2 provides a Report Builder you can use to build the following types of reports:

- TRENDview PRO Reports
- TRENDview PRO Data Set Reports (SDT)
- TRENDview HTML Reports

You can use the TRENDview PRO Client to modify and view your own reports or the ones in the TREND-supplied ReportPacks using a Java-enabled web browser.

The TRENDview HTML Client enables you to use any web browser to view reports that have been stored in HTML format.

These facilities are described in the following user manuals:

- *TRENDweb Installation and Administration Guide*
- *Guide to Building TREND Reports*
- *Guide to Viewing TREND Reports*

Managing Data

The Data Manager is an administrative tool that controls data rollup, allows aging criteria to be changed, and displays total database statistics. Aggregation and aging criteria can be viewed by all users, but only the trendadm user has the authority to change these criteria.

Once data has been collected, TREND manages the aggregation and summarization of the data automatically. You can use Data Manager to check the size of the database, manually purge data from tables, and control the retention times for specific data tables. The rate, hourly, daily, weekly, and monthly tables are the source for reporting.

TRENDcopy provides incremental, periodic, filtered database replication service to a backup server without the assistance of an experienced database administrator. Replication can be ad-hoc using a command-line interface or scheduled on a periodic basis. Database tables can be created and/or incrementally copied to another database based on user-definable criteria -- such as thresholds, time spans, or sets of nodes -- so that the new database contains only the information required for the reporting applications.

TREND is available with satellite database servers, allowing data to be collected and stored for a period of time at any physical location. Incremental, periodic data transfer can be triggered from a satellite location into a regional or central data repository. This action provides scalability by fully distributing the data collection and management throughout your enterprise. For example, you may wish to conserve WAN bandwidth by distributing polling agents and servers to remote locations, where data is periodically collected and managed in local repositories. Then, during off-hours, aggregated data can be moved to a more central database for reporting and analysis functions. The satellite database server package contains the satellite database server and a poller.

Advanced TREND features

TREND uses the concept of a "special" user. A user logged on as trendadm has privileged access throughout the TREND modules. For example, no user other than trendadm is allowed to set table aging criteria or to delete tables from the database. TREND is endowed with powerful features which trendadm can access for you. This manual focuses on the installation and use of TREND by the normal user. Some of the advanced features are listed here and described in detail in "TREND Configuration for trendadm" on page 2-1.

- Configuration
- Distributed Polling
- Multiple Databases
- Database and Partial Database Replication

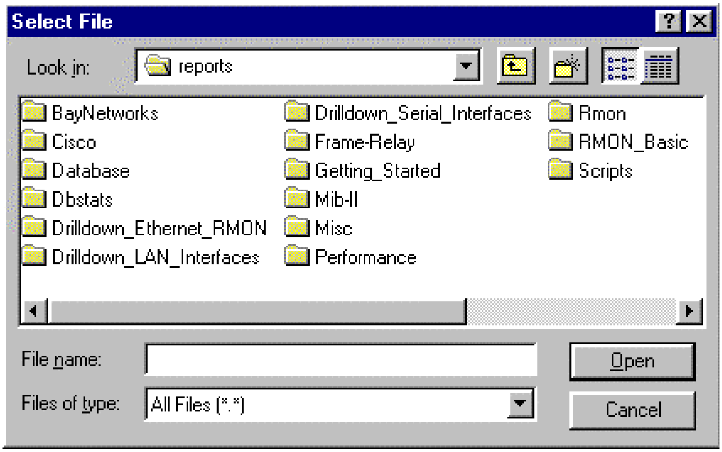
- Interface Key Descriptions

Commonly Used Commands

Some commands appear in multiple TREND application modules, for example The Open command appears in the File menu of the TRENDsheet, TRENDgraph, Grade of Service, and TRENDbuild modules. These common, repeated commands will be described here rather than repeating their descriptions in each chapter.

The Open Command

The Open command provides the user with access to the Select File window, where report query files can be viewed. To run a report by name, select the Open option under the File menu. This brings up the Select File window:



Look in: reports

File name:

Files of type: All Files (*.*)

Figure 1-2: The Select File Window

Initially, the Select File window points at your system's default report directory. If a report directory has been established through the Configuration menu, then this will be the default directory. This path is set as shown in "Changing the Default Report Directory" on page 2-11. If the report is in a different directory, you can get there by one of two ways:

1. Type the path name of the directory in the Filter box; then hit the Filter button.
2. Use the Directories box as a navigation aid. You can change directories by double clicking on an entry in this box, or clicking on an entry then clicking on the Filter button.

   The box has three types of entries:

   `.` The current directory (which is displayed in the Filter box). If you select this, you stay in the current directory.

   `..` The directory above the current directory. If you select this you will move up one level in the path name hierarchy. For example, if you are in `/usr/TREND/reports/Routers` and select `..`, you will move to `/usr/TREND/reports.`

   <subdirectory name> These are the names of directories below the current directory. If you select one of these, you will move down to the directory selected. For example, if you are in a directory called `/usr/TREND/reports,` and you select a value Routers from the Directories box, you will move to `/usr/TREND/reports/Routers.`

The Select File window shows you the names of all files in the current directory that match the value in the window's Filter box. You can use wild cards as part of the value on the line. By default, the value in the Filter box is *, so all files in the current directory are displayed.

Once you are in the directory in which the report you want to run is stored, you can run it by either double clicking on its name in the Files window, or clicking once on its name to select it, which will cause its name to appear in the Select File box, then clicking on the OK button.

The type of report defined in the file is determined by the file name extension, as follows:

Extension	Report Type
.gos	Grade of Service Report
.qgr	Graphical Report
.qss	Table (Spreadsheet) Report

Save

The Save option is used to save a previously saved report.

Save As

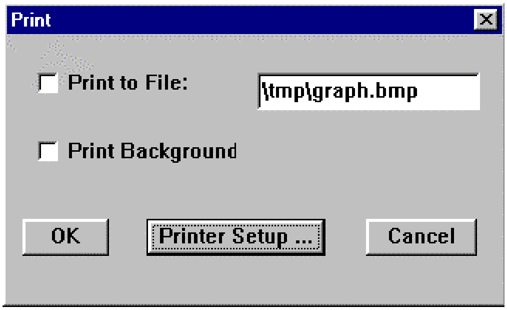
The Save As option is used to save a newly created file to disk, or to change the name of a previously saved report. Selecting the Save As command brings up the Select File window, which allows you to define the name and path of the file.

Print

By default, TREND uses postscript to provide a hard copy of the results of a report. The printout will give you the entire report as it appears in the display area (e.g., the area inside the box in a TRENDgraph window) on your screen. If there are multiple pages, they will be numbered.

To get a hard copy of a report, select the Print option under the File menu in the TRENDgraph, TRENDsheet, or Grade of Service windows. This action will bring up the Print window:

For Windows NT and Windows 95 systems, the following Print window will be seen:



□ Print to File: \tmp\graph.bmp

□ Print Background

Figure 1-3: The Print Dialog Box

In addition to postscript (the default), TREND also supports .bmp and .gif files. To save reports in these formats, click on the Print to File box and add the appropriate extension, either .bmp or .gif, to the file name. You can also modify the default file name to save the report file under any desired name.

Enter the name of the printer where you want the report run on the Printer line (for Windows, use the Printer Setup window). Be sure to send the report to a postscript printer for a postscript, .bmp, or .gif file. There is no default value for the printer. Enter the number of copies on the Number of Copies line. The default value is 1.

Report query files can be printed to a printer or a file with or without a background by clicking on the Print Background box.

- Click on the Print button to send the report to the printer.
- Click on the Reset button to change the values back to their original values.
- Click on Cancel to quit out of the window without printing.

Exit

The Exit command, found in the File menu of the Main TREND window is used to quit TREND. Similarly, the Exit command found in the File menu of all subsequent modules, lets the user quit that module and return to the Main window.

About TREND

To determine the version number of TREND running on your system, select About from the File menu in the menu bar of any TREND application.

TREND

2 TREND Configuration for trendadm

The TREND administrator, trendadm, has access to configuration and administration options not available to other TREND users. The trendadm can, for example, define users and views made up of different groups, nodes, and types to collect data on. This tailoring of the configuration is described in detail in this chapter.

TREND has advanced features that allow the trendadm to change defaults, or rollup and aging parameters of the data in the database. In addition, only the trendadm can truncate and delete tables in the database. The following sections describe these capabilities in more detail.

Configuring the System

The main TREND window allows the trendadm to configure the system and to take advantage of some of the advanced features of TREND, including distributed polling, setting user privileges, and utilizing multiple databases. You access these functions from the Configuration menu in the main TREND window. The main TREND window is displayed when you start TREND. Perform these steps to invoke TREND:

1. Log in as the trendadm user.
2. From the Start menu, select Programs>TREND. The main TREND window is displayed.

The Configuration menu has the following options:

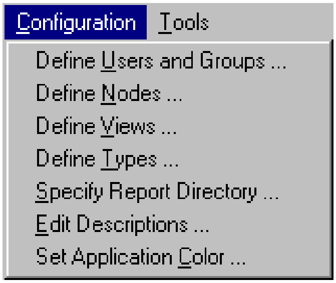


Figure 2-1: The TREND Configuration Menu

Nodes are organized into functional Types or Views. This organization is the basis for Collect Data's data collection setup process, where the nodes, node types, and node views defined in TREND determine which devices on your network are polled as a group for network performance statistics.

If you are not the trendadm user, the options in this menu are grayed out.

Defining Users and Groups

Users are TREND users you are registering with the application. Groups are collections of users who will share a common network node organization setup (see "Defining Nodes" on page 2-5 and "Defining Views" on page 2-7.)

To setup users and user groups, select the Define Users and Groups option under the Configuration menu. This will bring up the Define Users And Groups window:

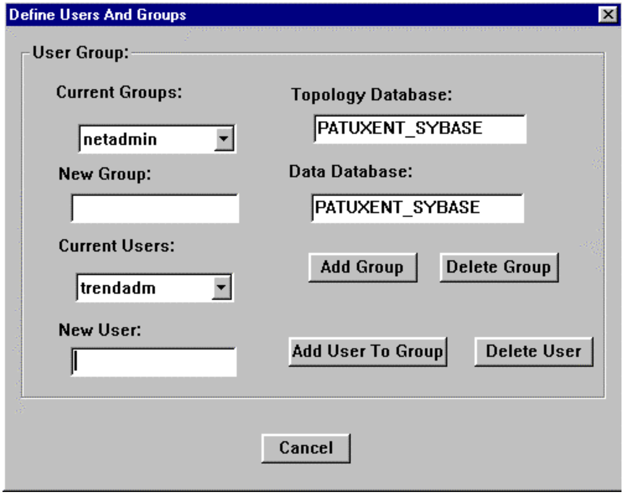


Figure 2-2: The Define Users and Groups Window

Follow these steps to create a new group:

1. Enter the name of the new group in the New Group box. The name can contain a maximum of eight characters. If you enter more than eight characters, the entry is truncated to eight characters.
2. Next, enter the name of the SQL Server which will store the group's node configuration information in the Topology Database box.
3. Enter the name of the SQL Server, which will store SNMP data collected through polling setup by a group member, in the Data Database box (see "The Sybase Interfaces File" on page 2-4).
4. Click on the Add Group button to register the group.
5. Click Cancel to quit out of the window.

The user will do the collecting. Follow the steps below to create a new user:

1. Enter the name of a user to be added by typing their UNIX system username in the New User box. Entries are case sensitive, and must exactly match an entry in the /etc/passwd file or in the NIS user database. The name can contain a maximum of eight characters. If you enter more than eight characters, the entry is truncated to eight characters.
2. Next, click on the downarrow of the Current Groups box and select the name of the group of which they will be a member.
3. Click on the Add User to Group button to register the user.

The Sybase Interfaces File

Sybase uses a file called interfaces to know where to find the SQL Servers on the network. On UNIX systems, this file is stored in the Sybase home directory named by your SYBASE environment variable (e.g. /usr/Sybase). Here is an example of an interfaces file:

```
PATUXENT_SYBASE      query tcp null-string patuxent 2025

                     master tcp null-string patuxent 2025

                     console tcp null-string patuxent 2026

AMUR_SYBASE          query tcp null-string amur 2025

                     master tcp null-string amur 2025

                     console tcp null-string amur 2026

JORDAN_SYBASE        query tcp null-string jordan 2025

                     master tcp null-string jordan 2025

                     console tcp null-string jordan 2026
```

The SQL servers named in this file are:

- PATUXENT_SYBASE
- AMUR_SYBASE
- JORDAN_SYBASE

The Sybase interfaces file on Windows NT systems is named Sybase\ini\sql.ini. Here is an excerpt from this file:

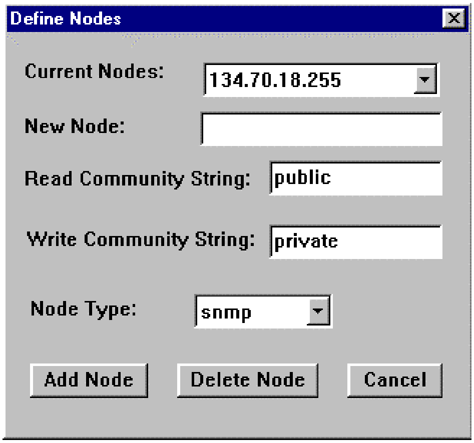
```
[SUWANEE]
master=NLMSNMP,\pipe\sybase\query
query=NLMSNMP,\pipe\sybase\query
master=NLWNSCK,suwanee,5000
query=NLWNSCK,suwanee,5000

[banyas]
master=NLWNSCK,banyas,2052
query=NLWNSCK,banyas,2052
```

The SQL servers in this file are SUWANEE and banyas.

The servers listed in the Sybase interfaces file can be entered on the Topology Database and Data Database lines in the Define Users and Groups window (see Figure 2-2, "The Define Users and Groups Window," on page 2-3). Entries are case sensitive, so make sure the value you enter exactly matches the entry in the Sybase interfaces file.

Defining Nodes

You can add nodes to the database through various methods. One method allows you to automatically search your subnets for the nodes using the Discover option from the TREND Main window. See "Discover" on page 5-1. Another method allows you to import a list of nodes from a text file. See "Import Nodes" on page 3-4. A third method allows you to add nodes, one at a time, through the Define Nodes window. To add a node, bring up the window by selecting the Define Nodes option under the Configuration menu:



Current Nodes: 134.70.18.255

New Node:

Read Community String: public

Write Community String: private

Node Type: snmp

Figure 2-3: The Define Nodes Window

- Click on the downarrow of the Current Nodes box to see a list of the nodes already entered. To see how one of these nodes is configured, select its name from the scrolling list of nodes in the Current Nodes box. Its values will be displayed in the various fields in the window, which are explained below.
- To add a new node to the database, enter its name in the New Node box. The name must be able to be resolved to an IP address through the /etc/hosts file, Domain Name Service or NIS.

SNMP GET and SET commands require passwords, known as community strings. The Read Community String is used for GETs, or polling, The Write Community String is used for SETs. The default Read Community String is public, and the default Write Community String is private. You can override the defaults for the current node by entering new values in the Read Community String and Write Community String boxes in the window. The strings are case sensitive and must match the strings expected by the device exactly.

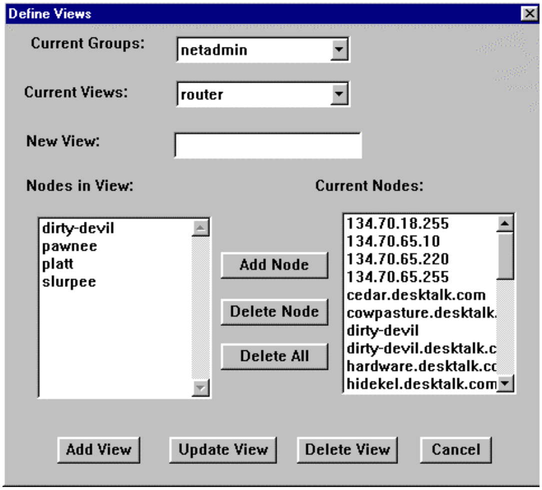
The Node Type is the kind of device you are entering (e.g., SNMP or RMON). Select the node type from the list available by clicking on the downarrow next to the Node Type box. You can poll groups of nodes that share a common type. Use the Define Types option from the Configure menu to develop a list of type values in advance to avoid inconsistencies (see “Defining Types” on page 2-9). RMON must be used only for RMON devices; all others are SNMP. Note that if you want to collect two kinds of data from a device, set it to RMON.

- To add a node once its properties are entered the way you want them, click on the Add Node button.
- To delete the current node, click on the Delete Node button.
- To quit out of the window, click on the Cancel button.

Defining Views

A *view* is a set of nodes that are grouped together so they can be polled for the same data by a single collection request. They may be grouped together because they share a common function, location, or other characteristic, which leads the user to want to monitor their performance the same way. Views can belong to different groups.

To create a view, select the Define Views option under the Configuration menu. This will bring up the Define Views window:



Define Views

Current Groups: netadmin

Current Views: router

New View:

Nodes in View:

dirty-devil
pawnee
platt
slurpee

Current Nodes:

134.70.18.255
134.70.65.10
134.70.65.220
134.70.65.255
cedar.desktalk.com
cowpasture.desktalk.
dirty-devil
dirty-devil.desktalk.c
hardware.desktalk.cc
hidekel.desktalk.com

Figure 2-4: The Define Views Window

- You can add a new view by typing its name in the New View box. You can then click on the nodes you would like to include in the new view. As you select a node it will become highlighted, hold down the shift key to select additional nodes.
- When you have selected all the desired nodes, click on the Add View button. The name you entered will be included in the Current Views list.

Note: To add another view, you must first delete the nodes from the Nodes in View box, then add new ones.

The list of existing views is displayed in alphabetical order in the Current Views scrolling list.

- To select an existing view, click on the downarrow of the Current Views box to bring up the list, then select the name of the view you want.

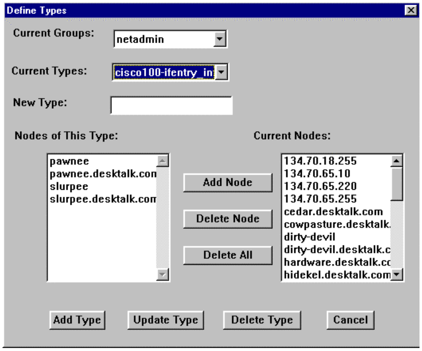
- ◆ To delete a view, select its name in the Current Views list and click on the Delete View button.

Note: To edit a view, you must delete the old view and create a new one using the methods described above.

- ◆ Click on the Cancel button to quit out of the window.

Defining Types

You can add types to the database through various methods. One method allows you to automatically identify the device types on your subnets by using Type Discovery from the command line. See "Running SNMP Type Discovery" on page 5-17. You can also create a type by selecting the Define Types option under the Configuration menu, which will bring up the Define Types window:



Current Groups: netadmin

Current Types: cisco100-ifentry_in

New Type:

Nodes of This Type:

pawnee
pawnee.desktalk.com
slurpee
slurpee.desktalk.com

Current Nodes:

134.70.18.255
134.70.65.10
134.70.65.220
134.70.65.255
cedar.desktalk.com
cowpasture.desktalk.
dirty-devil
dirty-devil.desktalk.c
hardware.desktalk.cc
hidekel.desktalk.com

Figure 2-5: The Define Types Window

- You can add a new type by typing its name in the New Type box. You can then click on the nodes you would like to include with the new type definition. As you select a node it will become highlighted, hold down the shift key to select additional nodes.
- When you have selected all the desired nodes, click on the Add Type button. The name you entered will be included in the Current Types list.

Note: Types can belong to different groups.

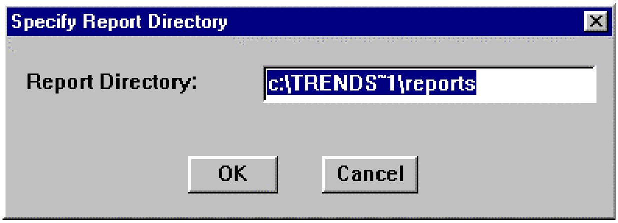
The list of existing types is displayed in alphabetical order in the Current Types scrolling list.

- To select an existing type, click on the downarrow of the Current Types box to bring up the list, then select the name of the type you want.
- To delete a type, select its name in the Current Types list and click on the Delete Type button.
- To edit a type, you must delete the old type and create a new one using the methods described above.
- Click on the Cancel button to quit out of the window.

Note: When adding another node, you must first delete the nodes in the Nodes of this Type box before adding new ones.

Changing the Default Report Directory

The default report directory is where TREND automatically looks for report definition files when the user selects the Open option under the File menu. The default report directory is $DPIPE_HOME/reports. Use the Specify Report Directory window to change the default path:



Report Directory: c:\TRENDS~1\reports

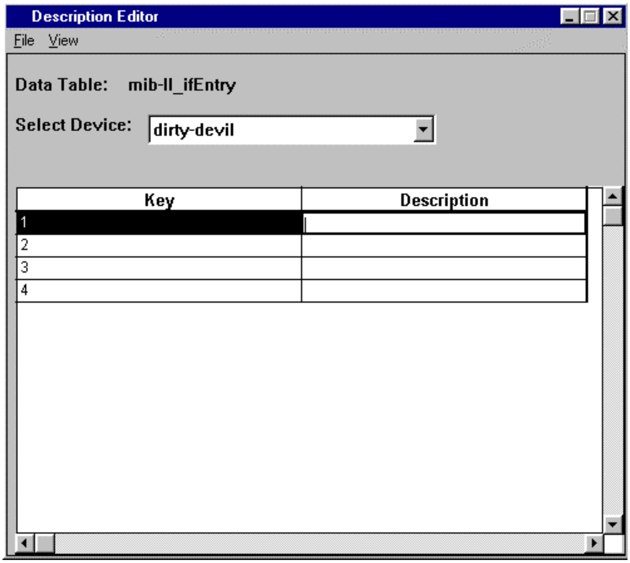
Figure 2-6: The Specify Report Directory Window

- To bring up the Specify Report Directory window, select the Specify Report Directory option under the Configuration menu. Enter the new path name in the Report Directory box and click on the OK button.

Typically, the system administrator establishes a central report directory as read-only for most users. It is suggested that $DPIPE_HOME/reports is used as the directory, with a separate directory setup for each copy of TREND software installed. Multiple directories can be established for common report access by different groups.

Defining Keys

Keys are selected using the Edit Descriptions command option from the Configuration menu. This action invokes the Description Editor window.



Data Table: mib-II_ifEntry

Select Device: dirty-devil

Key	Description
1	
2	
3	
4	

Figure 2-7: The Description Editor Window

Follow these steps to select a key:

1. Choose Select Table from the File menu
2. Click on a data table name in the Select Table window and click OK
3. Select a device from the Select Device list using the downarrow
4. Select Exit from the File menu

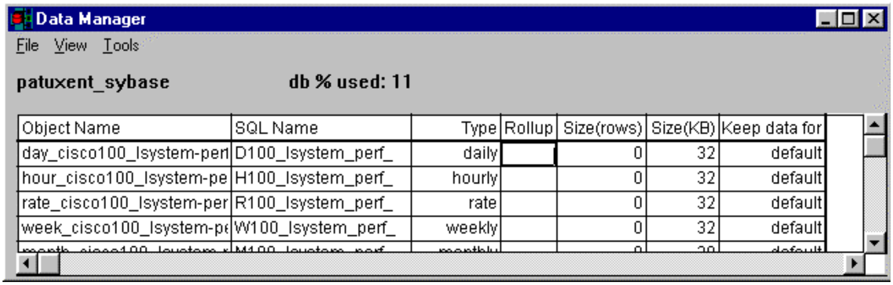
Setting Application Background Color

Users with color monitors can set the background colors for the MIBwalker, TRENDgraph, and Grade of Service windows through the Set Application Color option of the TREND Configuration menu. The Background Color window contains a palette of 64 standard background colors, plus sliders for infinite number of user-defined colors.

To use this application, simply select an application, select a color, and click on Apply. Then click on Cancel to Exit

Changing Data Rollup Parameters

TREND’s Data Manager function provides an easy way to read tabular display of information about every data table in your database(s), plus a central location for controlling data *rollup* and *aging*. Any TREND user can view information about the database via Data Manager. Only the administrative user, trendadm, can use Data Manager to change defaults or to modify aging and rollup for individual tables.



patuxent_sybase db % used: 11

Object Name	SQL Name	Type	Rollup	Size(rows)	Size(KB)	Keep data for
day_cisco100_lsystem-perf	D100_lsystem_perf_	daily		0	32	default
hour_cisco100_lsystem-pe	H100_lsystem_perf_	hourly		0	32	default
rate_cisco100_lsystem-per	R100_lsystem_perf_	rate		0	32	default
week_cisco100_lsystem-pe	W100_lsystem_perf_	weekly		0	32	default
month_cisco100_lsystem-r	M100_lsystem_perf_	monthly		0	32	default

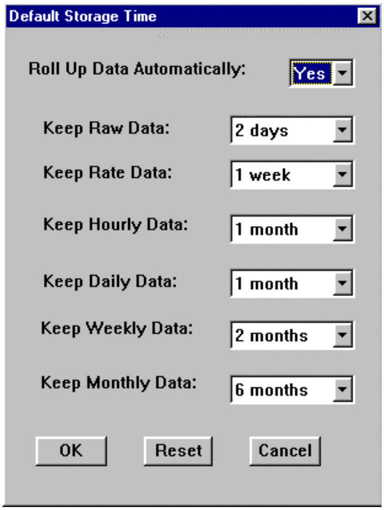
Figure 2-8: The Data Manager Main Window

Changing System-wide Rollup and Aging Defaults

You have the choice of allowing data to be aggregated and aged according to system-wide defaults or setting specific values on specific tables.

To set or view the system-wide default values, follow these steps:

1. Select Set Default Storage Time from the Data Manager’s File menu to bring up a window that shows you the current default settings:



Roll Up Data Automatically: Yes

Keep Raw Data: 2 days

Keep Rate Data: 1 week

Keep Hourly Data: 1 month

Keep Daily Data: 1 month

Keep Weekly Data: 2 months

Keep Monthly Data: 6 months

Figure 2-9: The Default Storage Time Window

2. Set the desired value for each field. To see the choices available, click on the downarrow button by the field you are setting. The fields to be set are:

   - Roll Up Data Automatically:

     Determines if sample data in a data table will be aggregated.

   - Keep <type> Data:

The period of time data in all tables of the type of table indicated will be kept. When a row has been in the table <save period + 1 days> it will be deleted from the table.

3. Click OK to save the newly set values.
4. Click Reset to re-display the values set the last time the OK button was clicked.
5. Click Cancel to exit.

Only the user trendadm can change the system default values. Other users will be able to change the values displayed in the window but will not be able to make the new values effective because the Apply button will not work (it is grayed out in the display).

Changing Rollup on a Specific Table

Change the value by typing a new one in the cell and hitting the return key. Valid values are:

- yes

  The data in this table will be rolled up.

- no

  The data in this table will not be rolled up.

- default

Whether the data in this table will be aggregated or not is determined by the value set for Roll Up Data Automatically in the Default Storage Time window (see “Changing System-wide Rollup and Aging Defaults” on page 2-14).

Only the user trendadm can modify a table's rollup option. For all others, clicking on the cell will simply highlight the entire row.

Changing Aging on a Specific Table

Change the value by typing a new one in the cell and hitting the return key. Valid values:

- 0/forever:

  Never discard the data in this table.

- <number>

  Actual number of days to keep a row of data in this table. A row will be deleted when it is <number + 1> days old.

- default

  The length of time data in this table will be stored is determined by the value set for Keep <*type*> Data in the Default Storage Time window, where <type> matches the Type column value for this table (see "Changing System-wide Rollup and Aging Defaults" on page 2-14).

Only the user trendadm can modify a table's *aging* option. For all others, clicking on the cell will simply highlight the entire row.

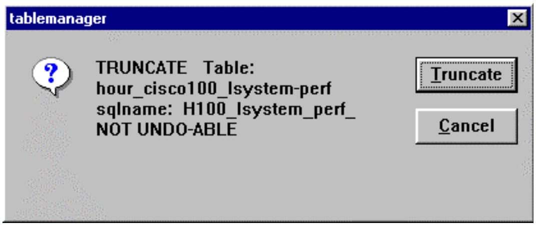
Truncating Tables

Truncating a table is when you delete of all rows in the table but keep the table itself. Truncating a table does not make any entries in the Sybase transaction log so it cannot be recovered.

Only the user trendadm can truncate tables. For all other users this option will be grayed out.

To truncate one or more tables:

1. Select the tables to be truncated by clicking on their rows in the Table Information area. The selected rows will be displayed in reverse video.
2. Select Truncate table(s) in the File menu.
3. For each table to be truncated, this window will be displayed:



TRUNCATE Table:
hour_cisco100_lsystem-perf
sqlname: H100_lsystem_perf_
NOT UNDO-ABLE

Figure 2-10: The Truncate Table Confirmation Message

The window shows the user visible name and the internal database name of the table to be truncated.

4. Click on the Truncate button to truncate the table.
5. Click on Cancel to keep the data in this table.
6. A new window will be displayed for the next table if more than one was selected.

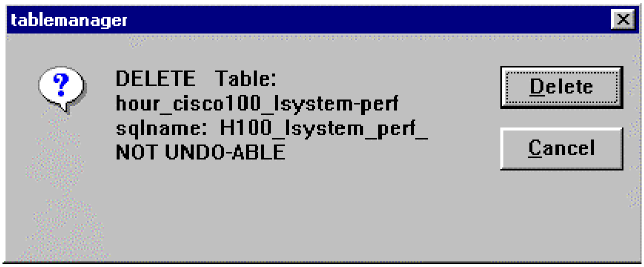
Deleting Tables

Deleting a table is when you "*drop*" (in database terminology) the table from the database. The table and everything in it are discarded. Deleting a table makes an entry in the *Sybase* transaction log so it can be recovered. However, if the transaction log space required to delete the table is larger than the space available, the delete will fail. If this occurs, truncate the table (see "Truncating Tables" on page 2-16); then delete it.

Only the user trendadm can delete tables. For all other users this option will be grayed out.

To delete one or more tables, follow these steps:

1. Select the tables to be deleted by clicking on their rows in the *Table Information* area. The selected rows will be displayed in reverse video.
2. Select Delete table(s) in the File menu.
3. For each table to be deleted, this window will be displayed:



DELETE   Table:
hour_cisco100_lsystem-perf
sqlname: H100_lsystem_perf_
NOT UNDO-ABLE

Figure 2-11: The Delete Table Confirmation Window

The window shows the user visible name and the internal database name of the table to be deleted.

4. Click on the Delete button to delete the table.
5. Click on Cancel to keep the table.
6. A new window will be displayed for the next table if more than one was selected.

License Key Administration

TREND uses a licensing scheme for copy protection. This section describes the license key process, including the use of the DSI_ELMHOST environment variable to assign a server host and improve performance.

TREND License Scheme

TREND uses a licensing scheme to prevent unauthorized usage. A license key controls all licensing attributes for each application module in TREND. A license key is a numeric string that resembles a rather long credit card number. Each key encodes the number of licenses (concurrent users) allowed for each group of TREND applications (also referred to as features or functions), as well as the start date and expiration date of the application, a server code, and the server's IP address.

Licenses are generally available to anyone on a network who can reach the license server running the license manager daemon. The license mechanism does not need to reside on the same server as the TREND application. These licenses, therefore, are known as floating licenses.

Example:

If 10 licenses are encoded in the license key, any 10 users on the network can use these licenses concurrently. When concurrent usage reaches 10, a subsequent request for a license will be denied because all licenses are in use.

Note: Desktalk recommends that the number of licenses be one greater than the number of simultaneous users, so that someone can always access the system as trendadm. Permanent keys are generated based on your system's IP address and hostid (a unique identifier for the CPU). If either of these changes, contact Desktalk for a new keys.

License keys in TREND are assigned to application modules based on their functionality as polling agents, rollup/aging processes, or client applications. These groups of applications and processes can be installed on a single machine, or on separate machines.

Function	**Installed on...**	**Applications**
Polling Agents	polling station	mw_collect rmon_collect*
Rollup/Aging Processes	client machine or database server	trendit db_delete_data
Client Applications	client machine licensed on an “X-term” or host/ user/ display basis	TREND TRENDbuild TRENDgraph TRENDsheet Grade of Service Data Manager MIBwalker TRENDcopy

TREND includes an additional licensing mechanism to track a purchased TREND database size against the database’s daily usage. Initially, you must call DeskTalk Systems Technical Support to obtain a license key for this new feature. If daily database usage exceeds the purchased database size, a daily warning message is displayed on the screen and in trend.log. You continue to receive these warning messages until you purchase an expanded TREND database and a new license key is generated.

License Composition

TREND uses a license key for copy protection and authorization. Each license key contains a composite of features. The TREND features, numbered from 30 through 40, are as follows:

- 30: Server
- 31: GUI Applications
- 32: Poller
- 33: GUI Applications (expert, i.e., MIBwalker)
- 34: Poller (RMON)
- 35: TRENDcopy
- 36: Unused at this time
- 37: Unused at this time
- 38: Database Size Verification (i.e., db_license)
- 39: Unused at this time
- 40: Unused at this time

Different TREND programs use the feature number to request a token from the license key. Tokens are generally available to anyone who can reach the license server. Clients (reporting and polling) are required to obtain a token from the correct location, specified by user environment settings. Once a license key is installed, the system issues a token authorizing the client to use TREND.

- Polling clients utilize either a local license key to obtain a token or must go to the server to obtain a token
- Reporting clients must go to the server to obtain a token

If there is an error message in the trend.log such as:

```
Server banyas [38]: No licenses are currently available.
```

Contact DeskTalk Systems Technical Support to have a new License Key generated that covers the updated range of expected tokens. This error is being generated by the *db_license* program which is expecting or using the *38* token for monitoring database size and, if still using a License Key generated for much earlier TREND releases, the updated range of tokens will not be available.

DSI_ELMHOST

When an application starts, it attempts to get a license from the license server on the network by sending out a broadcast. The application will search for a license on the network until it finds it or it times out. The license manager will normally locate the license server host automatically. You may explicitly specify the server host(s) via the environment variable DSI_ELMHOST. It is strongly recommended that the variable DSI_ELMHOST be set in the application environment (env) to the server where the license resides. This will allow better control of license server access and will also improve system performance. The files to be modified are Cshrc and Profile located in $DPIPE_HOME/lib.

Example:

If the license server resides on the host Desktalk:

`setenv DSI_ELMHOST Desktalk` (will broadcast)

`setenv DSI_ELMHOST @Desktalk` (will not broadcast)

Once DSI_ELMHOST is set, TREND applications request licenses from Desktalk as well as broadcasting a request on the local IP Subnet.

If there are multiple TREND installations where each is running a separate License Manager and users connect to each system directly (i.e., not using remote clients) to display TREND GUIs, then an implementation is available to provide further lockdown on license allocation by performing these steps:

1. Create a resource file.

UNIX Systems

a. Create a text file using VI and enter the following line:

`%IPACCEPT` *<IP Address of the License Host>*

b. Save the file in the directory **`/usr/trend_license`** with a name such as dsi_resource.

Windows NT

a. Create a text file using Notepad or Wordpad and enter the following line:

`%IPACCEPT` *<IP Address of the License Host>*

b. Save file in the directory **`%DPIPE_HOME%\lic`** with name such as dsi_resource.

2. Activate licensing with the resource file.

UNIX Systems

a. Edit the file **`/etc/rc.TREND_license_up`**.

b. After the end of line that starts **`/usr/trend_license/trend-elmd -m 0.25m`**, add the following phrase with a leading space:

```
-r /usr/trend_license/dsi_resource
```

c. Save the changes.

d. Stop Elan License Manager:

```
ps -ef | grep trend-elmd
kill xxxx
```

where *xxxx* is the process number shown in the ps listing.

e. Start Elan License Manager:

```
sh /etc/rc.TREND_license_up
```

Windows NT

a. Open the Control Panel.

b. Double-click on Elan License Manager.

c. Stop Elan License Manager.

d. Add pathname of file created earlier into 'Resource File:' prompt.

e. Click **OK**

f. Start Elan License Manager.

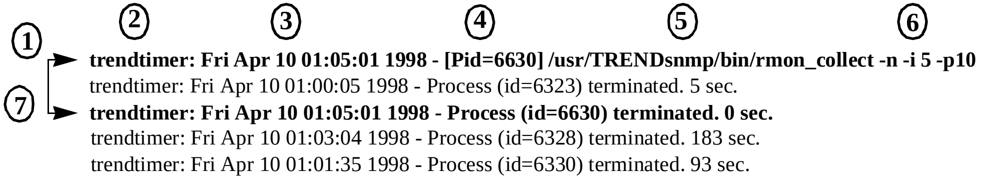
If you have any questions about this implementation, whether it should be applied in your enterprise, etc., please contact DeskTalk Systems Technical Support.

Message Handling

TREND uses logs to keep track of messages regarding the operation of the system. Messages appear in the log when processes begin and complete. Messages also appear when an error condition occurs. This section describes the basic format for the messages that appear in the log.

Example

The following example shows a portion of a log where processes initiate and complete. The highlighted items show two examples of processes initiating and completing; note that each pair has the same process id (Pid) number.



```
trendtimer: Fri Apr 10 01:05:01 1998 - [Pid=6630] /usr/TRENDsnmp/bin/rmon_collect -n -i 5 -p10
trendtimer: Fri Apr 10 01:00:05 1998 - Process (id=6323) terminated. 5 sec.
trendtimer: Fri Apr 10 01:05:01 1998 - Process (id=6630) terminated. 0 sec.
trendtimer: Fri Apr 10 01:03:04 1998 - Process (id=6328) terminated. 183 sec.
trendtimer: Fri Apr 10 01:01:35 1998 - Process (id=6330) terminated. 93 sec.
```

```
trendtimer: Fri Apr 10 01:10:00 1998 - [Pid=6668] /usr/TRENDsnmp/bin/mw_collect -n -i 5 -p10
trendtimer: Fri Apr 10 01:10:01 1998 - [Pid=6670] /usr/TRENDsnmp/bin/mw_collect -n -i 10 -p10
trendtimer: Fri Apr 10 01:10:02 1998 - [Pid=6671] /usr/TRENDsnmp/bin/rmon_collect -n -i 5 -p10
trendtimer: Fri Apr 10 01:10:02 1998 - [Pid=6672] /usr/TRENDsnmp/bin/rmon_collect -n -i 10 -p10
trendtimer: Fri Apr 10 01:05:03 1998 - Process (id=6629) terminated. 3 sec.
trendtimer: Fri Apr 10 01:10:02 1998 - Process (id=6670) terminated. 1 sec.
trendtimer: Fri Apr 10 01:10:02 1998 - Process (id=6671) terminated. 0 sec.
```

1. Process initiation message. This message shows when a process starts.
2. Scheduling program name. Trendtimer is the master scheduler program and it is associated with normal process initiation and process completion.
3. Date and time the process starts.
4. Process id number. The system assigns a process id number to each process.
5. Process name. File name and path of the process that was started by trendtimer.
6. Process options that are associated to the process specified in step # 5. (See “Man Pages” on page A-1or the appropriate chapter in this document for more information.)
7. Process completion message. This message shows when a process completes.

General Message Format

All messages specify the *program name* followed by the *date* and then the *message* in the format:

program name: date - message

An example of the complete log entry follows:

trendtimer: Fri Apr 10 01:00:02 1998 - Process (id=5678) terminated. 0 sec.

where:

program name	is	trendtimer
date	is	Fri Apr 10 01:00:02 1998
message	is	Process (id=5678) terminated. 0 sec

Every time trendtimer initiates a process (mw_collect, rmon_collect, trendit, lpr_launch, db_delete_data, etc.), an entry appears in the log. Periodic or regular processes appear every time the specific process is called such as at regular intervals of 5, 10, 15, 20, 30, or 60 minutes. An entry appears when the process initiates and when the process finishes with a process id number (Pid) attached to each entry.

Error messages appear in the same format as general messages. The following log entry shows an example of a process initiation message followed by a process error message and then followed by a process completion message.

```
trendtimer: Fri Apr 10 01:20:01 1998 - [Pid=6675] /usr/TRENDsnmp/bin/trendit -s
trendit: Fri Apr 10 01:20:01 1998 - Server net.xyz.com [30]: No licenses currently available.
trendtimer: Fri Apr 10 01:20:01 1998 - Process (id=6675) terminated. 1 sec.
```

Trendtimer initiates the process called trendit. Trendit displays an error message, and then trendtimer displays the termination message for trendit.

Process Initiation Messages

The format of the message portion that appears at process initiation is the following:

[Pid=*nnnn*] <*schedule file name*> <*options*>

where: *nnnn* is the process id number

<*schedule file name*> is the path name and file name associated with the specific process

<*options*> designate the parameters that modify how the process will run

For example, the format of the message portion that launches a collection request such as mw_collect or rmon_collect is the following:

[Pid=*nnnn*] <*schedule file name*>*-n* -i *xx* -p*yy*

where:	*nnnn*	is the process id number
	<*schedule file name*>	is the path name and file name associated with mw_collect or rmon_collect
	-*n*	indicates that data is collected for specified hosts
	-i *xx*	indicates the interval to repeat the poll
	-p*yy*	specifies the maximum number of objects that appear in the collection request

The following message shows a specific example for an mw_collect process initiation. Process 1234 collects data for mw_collect using a specified list of hosts at 5-minute intervals with a maximum of 10 SNMP variables per packet.

```
[Pid=1234] /usr/TRENDsnmp/bin/mw_collect -n -i 5 -p10
```

An example of the complete log entry follows:

```
trendtimer: Fri Apr 10 09:20:03 1998 - [Pid=1234] /usr/TRENDsnmp/bin/mw_collect -n -i 5 -p10
```

Process Completion Message

Every time a process completes normally, a message appears in the log. The format of the messages follows:

Process (id=*nnnn*) terminated. *x* sec.

where *nnnn* is the process id number and *x* is the number of seconds the process took to complete.

For example, the following message shows that process 5678 completed processing in less than 1 second.

```
Process (id=5678) terminated. 0 sec.
```

An example of the complete log entry follows:

```
trendtimer: Fri Apr 10 01:00:02 1998 - Process (id=5678) terminated. 0 sec.
```

Process Error Message

Any time a process produces an error, a message appears in the log with the date and timestamp the process records the message. The format of the message follows:

program name: date - message

The following example shows the log entry of an error message:

```
trendit: Fri Apr 10 01:20:01 1998 - Server net.xyz.com [30]: No licenses currently available.
```

TREND

3 The TREND Main Window

The Main window provides a point-and-click graphical user interface for access to TREND applications and reports. It is the control panel for TREND, invocable from a command line, or from any configurable command menu.

The Main window also allows the TREND system administrator, trendadm, to configure the system and to take advantage of the advanced features of TREND, including distributing polling, setting user privileges, defining interface descriptions, and utilizing multiple databases.

Note: You can get started using the basic features of TREND without custom configuration or trendadm expertise.

Display and Menu System

When you invoke TREND using the TREND executable, the Main window appears on your screen:

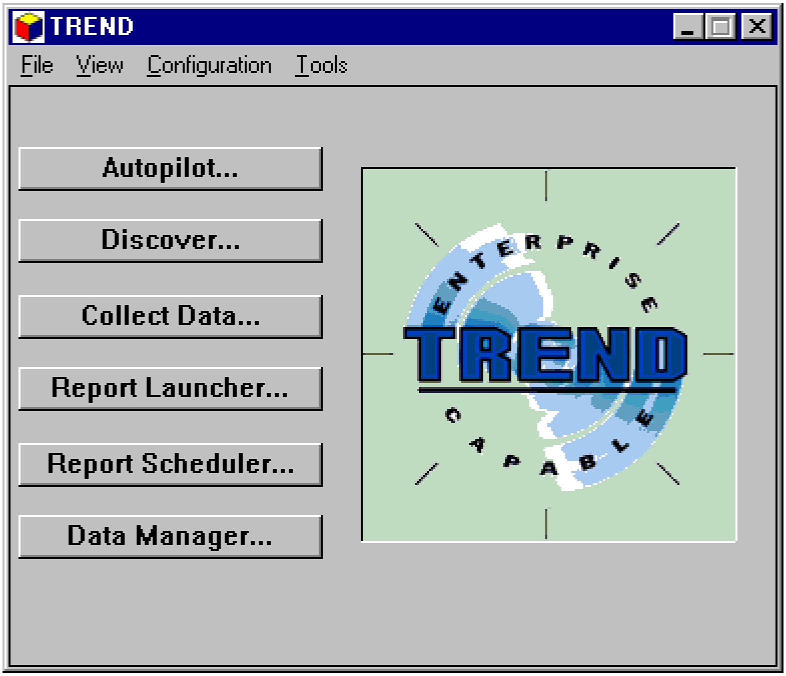


Figure 3-1: The TREND Main Window

CAUTION

In a Windows NT or Windows 95 environment, you can lose information on the TREND main window if Large Fonts is specified for the windows Font Size display settings option. Check the Font Size setting by selecting Settings from the Start menu, and then Control Panel/Display/Settings. Make sure the Font Size is set to Small Fonts.

The menu bar and application access buttons provide access to all TREND user visible GUI applications and functions. The buttons invoke the following applications, which are described in detail in their respective chapters in this guide:

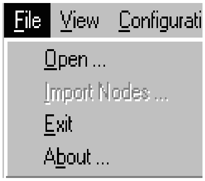
Application	**Chapter**
Autopilot	4
Discover	5
Collect Data	7
Report Launcher	14
Report Scheduler	15
Data Manager	8

All TREND application modules are brought up under your user name, with your environment. Some menu commands (e.g., table aging, truncation, and deletion from Data Manager) can only be executed by the trendadm user. Users who are not trendadm will see these commands grayed-out indicating that they are unavailable.

Menu Command Summary

The menu bar has File, View, Configuration, and Tools pull-down menus.

The File Menu



The File menu contains four options: **Open, Import Nodes, Exit,** and **About.** Open, Exit, and About are all standard commands, and so are described in "Commonly Used Commands" on page 1-15.

Import Nodes

The **Import Nodes** command allows the user to import nodes, views, and types via an ASCII file. To utilize this feature, select Import Nodes from the File menu of the main TREND window. A dialog box will request the name of the node, view, or type file. Enter the name and click OK.

The format of the imported file must be:

```
node_name, read_community, write_community, poll_type,
type_name, view_name, group_name
```

Required fields are **node_name, read_ community, write_community,** and **poll_type** (snmp or rmon). If **type_name** or **view_name** is specified, **group_name** must also be specified. Missing fields in a record must be represented by the comma delimiter.

You can also issue the import nodes command from the command line. The syntax is:

import_nodes -f *import_file* **-h -V**

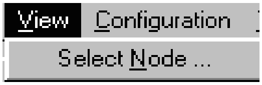
where:

-f Reads *import_file.*

-h Displays the command line options (help).

-V Displays the current version of the import_nodes utility.

The View Menu

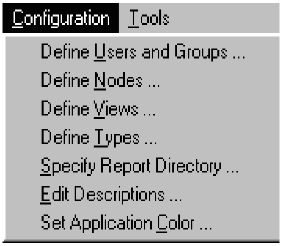


The View Menu contains one option, **Select Node.**

Select Node

Select Node is used to set the initial target node value for polling. This value applies to the MIBwalker tool and the Collect Data application module. Other targets can be set within those modules.

The Configuration Menu



Configuration menu options allow the user to define users, groups, views and types for set-up and configuration of the system to be reported on. Command options in the Configuration Menu are only available to your TREND administrator, *trendadm*. See "TREND Configuration for trendadm" on page 2-1 for a complete description.

Define Users and Groups

Define Users and Groups allows *trendadm* to set up users and groups (see “Defining Users and Groups” on page 2-2).

Define Nodes

Define Nodes adds nodes to be polled to the database (see “Defining Nodes” on page 2-5).

Define Views

Define Views creates a view in TREND. A view is a set of nodes that are grouped together so they can be polled by a single collection request (see “Defining Views” on page 2-7). A view is used to group together nodes on a geographic or organizational basis.

Define Types

Define Types associates a type with a group of nodes. A Node Type is the kind of device (e.g., router, RMON, etc...), and allows you to poll nodes collectively that share a common type (see “Defining Types” on page 2-9).

Specify Report Directory

Specify Report Directory changes the default path for the report definition directory (see "Changing the Default Report Directory" on page 2-11). The default path is **$DPIPE_HOME/reports.**

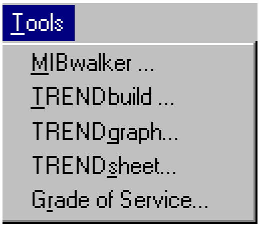
Edit Descriptions

Edit Descriptions adds or modifies descriptions for interfaces associated with specific device tables (see "Defining Keys" on page 2-12).

Set Application Color

Set Application Color sets the background color for the MIBwalker, TRENDgraph, and Grade of Service applications. The user must have a color monitor to use this function (see "Setting Application Background Color" on page 2-13).

Tools



The **Tools** menu contains two options for building and defining variables and reports:

- Select MIBwalker to define new groups of MIB variables to poll for, or to browse through the MIB tree structure using a point-and-click interface (refer to "MIBwalker" on page 6-1, for more detailed information).
- Select TRENDbuild to select variables, assign custom aliases, and build expressions that are written to a query file used by TRENDgraph or TRENDsheet (refer to "TRENDbuild" on page 10-1, for more detailed information). TRENDbuild can also be directly accessed through TRENDgraph or TRENDsheet.

You can also invoke the following reporting facilities from the Tools menu:

- TRENDgraph (see "TRENDgraph" on page 12-1)
- TRENDsheet (see "TRENDsheet" on page 11-1)
- Grade of Service (see "Grade of Service" on page 13-1)

4 Autopilot

Autopilot is used for installing and removing TREND ReportPacks. Please see the *ReportPack Guide, Chapter 1* for more information.

TREND

5 Discover

This chapter describes the TREND Discover tool, which can be used to get information about the nodes on your network.

The TREND Discover tool allows you to conduct a search that can:

- Find the nodes on your system
- Ascertain whether or not each node is SNMP manageable
- Determine what kind of device (router, hub, switch, etc...) the node is
- Automatically update the tables that control data collection.

As a result, Discover gives you the option of using an automated process to define your network.

Note: The use of Discover is optional. You can also populate and update your various node tables manually, if you choose.

What Discover Does

Discover can conduct two different kinds of searches. It pings a range of IP addresses, one at a time, and then analyzes and identifies the responses it gets. It uses that information to populate or update tables in your TREND database.

There are three specific pieces of information Discover can determine from this process:

- Whether or not there is a device at a specific IP Address
- If so, whether or not that device is SNMP manageable
- If a device is SNMP manageable, what kind of device it is.

One kind of TREND Discover process, IP Discovery, verifies the existence of a device at a particular IP address, and identifies whether or not that device is SNMP manageable.

The other Discover process, SNMP Type Discovery, ascertains what kind of device the node is.

Note: If you want information regarding an understanding of how Discovery interacts with the schema and expanding the list of devices that TREND discovers, see “Discovery Considerations” on page E-1**.**

IP Discovery

IP Discovery is a manually initiated process. It can be initiated either from the Discover user interface, or from the command line. The purpose of IP Discovery is to find the devices on a network, and determine whether or not they are SNMP manageable.

Note: Any TREND user can initiate Discover, but only from a TREND server. On TREND Clients, the Discover tool is disabled.

How IP Discovery Works

You initiate an IP Discovery by defining a range of IP addresses, a Subnet mask, and one or more read community strings to be searched. Discover then pings each IP address (except beginning subnet addresses (i.e., 0) and broadcast addresses (i.e., 255)) in that IP address range.

- If there is no device at the IP address, there is no response.
- If there is a device at that IP address that recognizes the ping, it responds to Discover. From this response, Discover knows that there is a device at that IP address. Discover then:
  - Sends an SNMP GET message to the responding device.

    If the device recognizes and responds to the GET, Discover identifies the device as SNMP manageable.
  - Attempts to discover the host name of the device (if you have specified host name translation).

    If your system has a protocol to translate IP addresses to host names, and the host name is found, Discover associates it with the discovered device. If the host name is not found, Discover lists the node by its IP address.

IP Discovery Table Population

When Discover gets a response from an SNMP capable device, it inserts the host name (or IP address) of the device into the dsi_nodes table. Once it exists in dsi_nodes, the discovered device is accessible to TREND.

Note: TRENDdiscover only places primary IP addresses, or hostnames if using translation, into the dsi_nodes table. You may use an ISQL session to reference the dsi_node_list table to determine the TRENDdiscover analysis of primary versus secondary addresses. The `dsi_primary` **and** `dsi_macindex` **columns in the dsi_nodes table contain the addresses.**

The dsi_nodes table also has a column that defines whether or not an SNMP device is RMON manageable. However, IP Discovery cannot determine this. RMON capability is discovered by SNMP Type Discovery. When Discover finds a new SNMP capable node, it inserts it into dsi_nodes as an SNMP device by default.

SNMP Type Discovery

Note: SNMP Type Discovery is scheduled by default to run once a day. In addition, when you install a TREND Report Pack, the Autopilot asks if you want to run SNMP Type Discovery at that time.

SNMP Type Discovery is typically run automatically, usually once a day. It can only be initiated in two ways:

- By the TREND Autopilot, when you use it to install a TREND ReportPack
- From the command line

Note: The Discover GUI accessed from the TREND Main window is only used to launch IP Discovery.

The purpose of SNMP Type Discovery is to identify the nature of the devices in the dsi_nodes table.

In particular, SNMP Discovery determines the Type of a discovered device. SNMP Type Discovery interrogates nodes to determine what kind device they are, and records the information in the TREND database. This identification allows TREND to target a specific type of device for data collection.

How SNMP Type Discovery Works

Discover uses special files, identified by the extension .dis, to determine whether or not a device is of a certain type. Each .dis file contains the necessary protocols to test a node for a single, specific type.

Note: .dis files are discussed at greater length in "Running Discover from the Command Line" on page 5-11**.**

In SNMP Type Discovery, the Discover tool reads the dsi_nodes and dl_type tables to identify the nodes that require type validation.

SNMP Type Discovery Table Population

There are two database tables that are affected by SNMP Type Discovery, dl_type and dsi_nodes.

For each node on your network, the **dl_type** table identifies the type of device it is. SNMP Type Discover populates this table by interrogating each new node in dsi_nodes, and adding an entry to dl_type.

The **dsi_nodes** table contains a row for each recognized node on your system. In addition, one column of dsi_nodes identifies whether a node is SNMP or RMON (default is SNMP). When SNMP Type Discovery identifies an RMON-manageable device, it updates this column of dsi_nodes.

See “Discovery Considerations” on page E-1 for additional technical information about the .dis files and type discovery.

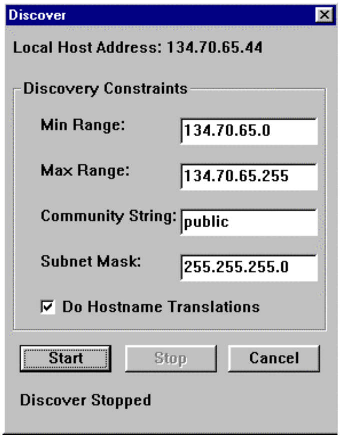
Using the Discover Graphical User Interface

TREND provides a Graphical User Interface (GUI) for its Discover tool. This section describes how to use this GUI to run IP Discovery. Only IP Discovery can be run using the Discover GUI. SNMP Type Discovery must be run from the command line

Note: IP Discovery can also be run from the command line. Instructions for how this may be done are provided in “Running Discover from the Command Line” on page 5-11.

Launching the Discover Interface

The TREND Main window includes a button to launch the Discover interface. When you click the Discover button, the Discover tool appears, as shown below:



Discover

Local Host Address: 134.70.65.44

Discovery Constraints

Min Range: 134.70.65.0

Max Range: 134.70.65.255

Community String: public

Subnet Mask: 255.255.255.0

☑ Do Hostname Translations

Discover Stopped

Figure 5-1: The TREND Discover Interface

Note: If you have the TREND Autopilot module open, you will need to close it before you can launch Discover.

The Start, Stop, and Cancel Buttons

There are three command buttons in the Discovery Interface: Start, Stop, and Cancel. The function of these buttons is described below:

Start

The Start button initiates the discovery process. The Start button is only available when Discover is stopped, as shown by the legend Discover Stopped at the bottom of the window.

Stop

The Stop button halts the discovery process. The Stop button is only available when Discover is running, as shown by the legend Discover Running at the bottom of the window.

Cancel

The Cancel button closes the Discover window. It does not affect the running process in any way.

Note: If you click the Cancel button while Discover is running, the window closes, but Discover CONTINUES TO RUN. If you want to stop the Discover process, you must use the Stop button.

Defining an IP Discovery

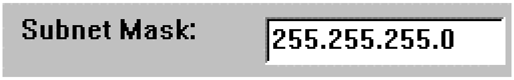
Discover also allows you to define the parameters of your discovery. The parameters that can be defined are:

- Subnet Mask
- IP Address Range
- Read Community Strings
- Host name Translations (on/off)

These parameters are briefly described in the following sections.

Subnet Mask

When the Discovery window appears, the Subnet mask is set to eight bits by default, as shown below.



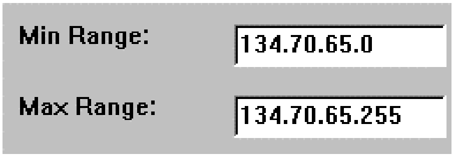
Subnet Mask: 255.255.255.0

If the Subnet you want to search requires a different Subnet mask, you can change the mask using this text field.

Note: The Subnet Mask you apply determines the limits of the IP Address range you can define for discovery.

Setting a Range for Discovery

Discovery allows you to specify a range of IP addresses to discover. You do this by specifying a minimum and maximum value, using the fields shown below.



Min Range: 134.70.65.0

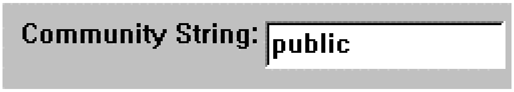
Max Range: 134.70.65.255

Note: The range you can discover depends upon your Subnet Mask. You must ensure that the Subnet Mask field is set appropriately for your network’s Subnet Mask

- If the discovery you want to perform is in your local Subnet, simply set the Min and Max Ranges, and click Start.
- The Discover interface appears showing the IP Range of your machine’s local Subnet. If you are running Discovery from a machine that is not part of the target Subnet, you will need to reset the MIN and MAX Range values to discover the target Subnet.

About Community Strings

When Discover finds a device at an IP address, it sends an SNMP GET to determine whether or not that device is SNMP manageable. This SNMP GET includes a Read Community String. The targeted device will only respond to Discover if its community string matches the string in the GET. For this reason, the Community String field (shown below) is an important one.



Community String: public

In the Discover GUI, you can only specify one community string at a time. The discovery process can only identify SNMP manageable nodes that have this community string.

Note: When IP Discovery is run from the command line, it has an option that allows you to discover for multiple community strings. See “Running Discover from the Command Line” on page 5-11**.**

To change a Community String, select the character string that appears in the Community String text field (by default, Public), and overwrite it.

Host Name Translations

Host names are (typically) English language names used to identify devices. If you select the Do Hostname Translations option (shown below), Discover will attempt to find the host name for the IP address of each node it finds.

☑ Do Hostname Translations

Different systems use different protocols to perform this translation. Discover will try to use whichever protocol is active on your system. If the host name is discovered, Discover uses it to identify the device. If no host name can be found for the target device, Discover uses the IP address.

Running Discover from the Command Line

Although there is a GUI for running IP Discovery, this process can also be run from the command line. SNMP Type Discovery can only be run from the command line.

Using the Command Line to Run IP Discovery

As described in "IP Discovery" on page 5-2, IP Discovery pings to discover nodes in the range you specify. When it finds an IP address that has a device, IP Discovery sends an SNMP Get to determine whether or not that device is SNMP manageable. This process can be accomplished from the command line.

The syntax is:

```
trend_discover   [-a <delete age> (default 10) ]
                 [-c <community string for all SNMP GETs> ]
                 [-C <ping number of packets (1 default)> ]
                 [-d <debug level> ]
                 [-D (suppress domain names) ]
                 [-E (echo progress) ]
                 [-f <community names file> ]
                 [-F <type definition file> ]
                 -h <end IP range>
                 [-H (help) ]
                 -l <start IP range>
                 [-o <SNMP timeout in seconds> ]
                 [-O <ping timeout in ms (NT ONLY) default 750ms> ]
                 [-p <max entries per pdu (default 20)> ]
                 [-P <SNMP port number (default 161)> ]
                 [-r <SNMP retries> ]
                 -s <network Subnet>
                 [-S <ping packet size (default: 1 byte)> ]
                 [-t (do type discovery) ]
                 [-u <username> ]
                 [-V <display version> ]
                 [-z <activate host translation> ]
```

Of the options described above, only -l, -h, and -s are mandatory:

```
trend_discover -l <start IP range> -h <end IP range> -s <network
Subnet>
```

Option Descriptions

-a	Specifies how many consecutive attempts to ping a node must fail before the node is automatically deleted from the dsi_nodes and dl_type tables.
-C	Specifies the number of packets that can be in a discovery ping. The default is 1.
-c	Specifies the single community string to use for all SNMP GETs.
-d	Specifies the debug level.
-D	Suppresses domain names.
-E	Causes the progress of Discover to be displayed on the screen.
-f	Specifies that Discover will read a community strings file and use the character strings in that file for its SNMP GETs.
-F	Name of the type definition file. Can only be used in conjunction with the -t option.
-h	Defines the maximum value of the discovery range.
-H	Causes a list of available Discover command line options to be displayed on your screen.
-l	Defines the minimum value of the discovery range.
-o	Determines how long Discover will wait before timing out an SNMP GET request.
-O	Specifies the time in milliseconds that must pass before Discover times out a ping when Discover is running from a Windows NT platform. The default is 750 milliseconds.
-p	Resets the maximum number of entries that can be in a Protocol Data Unit (pdu). The default is 20.
-P	Defines the port number to be used. The default is 161.

-r Defines the number of times Discover will retry its SNMP GET if there is no response.

-s Defines the Subnet Mask to be used in the Discover process.

-S Specifies the packet size of a discovery ping. The default is 1 byte.

-t Causes Discover to run an SNMP Type discovery.

-u Defines the TREND user name of the user conducting the discovery.

-V Displays the version number of the product.

-z Specifies that Discover will attempt to translate a discovered IP address to a host name.

Notes About Some Discover Command Line Options

- If the -u <username> option is not used, Discover will try to get a user name from the USER environment variable. If Discover cannot locate a USER environment variable, trendadm is used for the username.
- If you use either the -f or -F option (or both), giving a file name without specifying a file path, Discover will look for the specified file in the $DPIPE_HOME/scripts directory.
- If you use the -t option to run SNMP Type Discovery but do not use -F to specify a .dis file, Discover will look for .dis files in the DPIPE_HOME/scripts directory. If the .dis files are missing, an error message appears.
- The -a (delete age) option specifies how many consecutive attempts to ping a node must fail before the node is automatically deleted from the dsi_nodes and dl_type tables.
- For the Start IP range (-l), end IP range (-h) and network Subnet (-s) options, if the value you input is not complete, Discover will append zeros to the end of that value. For example, if you input the value *134.70*, Discover will assume that input to be 134.70.0.0

- When you use the -D (suppress domain name) option, translated host names will be truncated after (including) the first period in the host name.

Note: When the node translation (-z) option is on and a ping is successful, Discovery will perform a hostname lookup. A previously discovered translated name can only be replaced by another translated name, not an IP address. In other words if the host name lookup fails to return translated name, Discover will not replace the current name with an IP address. If the host's name is not valid anymore, the translated name has to be updated manually, and node name entries in dsi_snmp_nodes, dsi_nodes, dl_type and mw_collection tables must also be manually updated.

CAUTION

If a node is defined via Define Nodes or any other application, then MAC address information for this node is not available to Discovery. This can cause duplicate polling.

Specifying Community Strings for an SNMP GET Request

When Discover performs an SNMP GET request, the community strings it uses can be defined in the following ways:

- If you do not specify a community string, Discover will use the default, which is public.
- If you want to specify a single community string for all the SNMP GETs in a discovery, you can use the -c `<community_string>` option.

  Whatever character string you input with the -c option will be used as the identifying community string for the GET requests.
- If you want to specify multiple community strings for the SNMP GETs, you can use the -f option.

  This option can be used in two ways:

- If you use the syntax -f <`file_name`>, where `file_name` is the name of a file, but does not include the file's path, Discover will look for that file only in the $DPIPE_HOME/scripts directory.
- If you use the syntax -f <`file_path`>, where `file_path` is the name and path of a file (including either / or \), Discover will look for that file only in the specified path.

This file is a regular ASCII text file, with one read community string per line. Discover will try the community strings in this file until the first successful SNMP request is performed.

Community Strings Files

Discover uses the entries in the Community Strings file in the order in which they appear. The sequence is as follows:

1. Discover polls the first node, using the strings identified in the Community Strings file.

   Discover polls using the first string. If there is no response, Discover tries the second string, then the third, etc.

   When the node responds to a poll, Discover classifies that device as responding to that community string, and goes on to the next node.
2. Discover does the same for each node.

It is up to you to determine in which order the community strings appear in the Community Strings file.

CAUTION

In some cases, it is possible for devices on a network to respond to multiple community strings.

Example:

Some RMON vendors allow the use of public as the read community string for MIB-II tables, but require different community strings for RMON tables. This community string can also be used to GET MIB-II tables. In this case, the community strings file should list the RMON community string above the public community string.

Note: If a node is discovered with a different community string than the one it had when it was originally discovered, its community string will only be updated if the original community string was public.

When you create a Community Strings file, we recommend that you list the more common community strings first, as this can speed up the discovery process. However, you should keep in mind the possibility that your system may include devices that respond to multiple community strings, as described above.

Running SNMP Type Discovery

You can initiate SNMP Type Discovery in two ways:

- By the TREND Autopilot, when you use it to install a TREND ReportPack
- From the command line

This section describes running this kind of discovery from the command line.

Note: When you use the TREND Autopilot to install a TREND Report Pack, the Autopilot will prompt you to run SNMP Type Discovery, and will launch the program for you if you choose. At this time, there is no GUI that allows you to define SNMP Type Discovery.

Launching SNMP Type Discovery

SNMP Type Discovery is a variation of the Discover tool, which is launched from the command line as trend_discover. To launch SNMP Type Discovery, use the syntax:

```
trend_discover -t
```

This starts the Discover program, and specifies an SNMP Type Discovery.

SNMP Type Discovery Command Line Options

The only valid command line arguments for SNMP Type Discovery are:

```
trend_discover    [-d <debug level>]

                  [-H (help)]

                  [-p <max entries per pdu (default 20)>]

                  [-P <snmp port number (default 161)>]

                  [-r <SNMP retries>]

                  [-t (do type discovery)]

                  [-u <username>]

                  [-E (echo progress)]

                  [-F <type definition file>]

                  [-o <SNMP timeout in seconds>]

                  [-V <display version>]
```

Option Descriptions

-d Specifies the debug level.

-E Causes the progress of Discover to be displayed on the screen.

-F Names the type definition file. Can only be used in conjunction with the -t option.

-H Causes a list of available Discover command line options to be displayed on your screen.

-o Determines how long Discover will wait before timing out an SNMP GET request.

-p Resets the maximum number of entries that can be in a Protocol Data Unit (pdu). The default is 20.

-P Defines the port number to be used. The default is 161.

-r Defines the number of times Discover will retry its SNMP GET if there is no response.

-t Causes Discover to run an SNMP Type discovery.

-u Defines the TREND user name of the user conducting the discovery.

-V Displays the version number of the product.

Type Discovery Files

When you execute an SNMP Type Discovery, trend_discover tries to assign different types to the nodes specified in dsi_nodes. Once a node is determined to be of a particular type, it is inserted into the dl_type table. SNMP Type Discovery is driven by information defined in Type Discovery Files. The path of a Type Discovery File can be specified via the -F option (see notes above). By default, trend_discover searches for *.dis files in the $DPIPE_HOME/scripts directory. If .dis files are missing, an error message appears. Either install a ReportPack or create a custom report with a corresponding .dis file before running Type Discovery again.

Note: Once a node is determined to be of particular type, trend_discover will not try to rediscover that node for the same type again.

The valid keywords for Type Discovery Files are the following:

Keyword	Syntax	Description
typename	**typename:***name_of_type***;**	Assigns a name to a type. typename:cisco_routers;
collection	**collection:** *y/n***;** where *y/n* is **yes** or **no**. The default is **no**.	Specifies whether or not the mw_collection table controls Type Discovery. If the value is **yes**, then trend_discover performs type discovery only if the mw_collection table has an entry where username is the user running trend_discover, the device_type is the *name_of_type*, and the view_name is not set.
nodetype	**nodetype:***collect_type***;** where *collect_type* is **rmon** or **snmp**. The default is **snmp**.	Specifies whether this is an RMON or regular SNMP table.

The syntax rules for Type Discovery files follow:

1. Every line in Type Discovery files must end with a semi-colon (;), except for comments.

2. Comments begin with a slash and an asterisk (/*) and end with an asterisk and a slash (*/), as shown in the following example:

   /* this is a comment */

3. Blanks are ignored in the input file, unless they are inside a string (between single quotes), as shown in the following example:

   ‘ this is a string ’

SNMP Tests

The actual definition of the type defines one or more SNMP Tests that a device must pass in order to be of the given type. If multiple SNMP Tests are defined for a given type, all of them must pass (i.e., AND function is used). The following tests are available:

Test	**Syntax**	**Description**
Miblet	*miblet_name*; where *miblet_name* is the alias name defined in dsi_tab_alias.	This test checks that the device supports the miblet specified, to belong to the type. Example: mib-II_ifentry;
Simple Oid	*oid*;	To pass this test, the device must support the oid. The value returned by the device is don't care. Example: 1.3.6.1.2.1.1.1;
Simple NOT Oid	~*oid*;	To pass this test, the device must NOT support the oid. Example: ~1.3.6.1.2.1.1.1;

Test	Syntax	Description
Value	**oid = ‘***string_value***’;** or **oid != ‘***string_value***’;** or **oid =** *numerical_value***;** or **oid !=** *numerical_value***;**	To pass this test, the device must support the given oid and the return value must satisfy the defined expression. Example: 1.3.6.1.2.1.1.1 = ‘cisco router’; /* this is regular string compare, case sensitive */ Example: 1.3.6.1.2.1.1.1 = ‘*cisco*’; /* this is wild character compare, when wild characters are used trend_discover will perform case insensitive compare */ Example: 1.3.6.1.2.1.1.1 != ‘bay router’; /* this is not equal test */ Example: 1.3.6.1.2.1.1.1=453;

Input File Examples

Example 1

```
/* this file defines 3com_device type */
/* the test is done on sysDescr oid */
typename:3com_devices;
collection:no;
1.3.6.1.2.1.1.1=‘*3com*’;
```

Example 2

```
/* this file defines 3com_routers */
/* the test is done on sysDescr oid and ipForwarding oid */
typename:3com_router;
collection:no;
1.3.6.1.2.1.1.1=‘*3com*’;
1.3.6.1.2.1.4.1=1;
```

Example 3

```
/* this file will define types that support mib-II_ifEntry miblet */
/* and have ipForwarding turned on */
typename:mibII;
collection:no;
mib-II_ifEntry;
1.3.6.1.2.1.4.1=1;
```

Example 4

```
/* this file will define rmon1 ether stats type */
/* this type is limited to rmon1 devices that do not support rmon2 extioins */
typename:rmon1ethstats;
collection:no;
nodetype:rmon;
~1.3.6.1.2.1.16.1.4.1.1;  /* doesn’t support rmon2 extension defined for rmon1 */
1.3.6.1.2.1.16.1.1.1.1;  /* must support etherStatsIndex oid */
```

TREND

6 MIBwalker

The SNMP protocol supports management of network resources through a data structure called a Management Information Base (MIB). MIBs are composed of groups of managed objects, which are tables of variables that can contain a node’s statistical values and configuration parameters. A *node* is used here to mean any network communications accessible device that can be polled using SNMP. Each node supports the MIBs the node manufacturer feels are appropriate to the function of the node. Definitions for standard MIBs appear in Request for Comment (RFC) documents from the Internet Engineering Task Force. Definitions for enterprise MIBs come from the manufacturers of the nodes.

Use MIBwalker as a tool to perform the following functions:

- Learn about the contents of a MIB file
- Select sets of managed objects to define data collection tables
- Conduct GET and GET NEXT requests to ensure that devices can be polled

When you execute MIBwalker it displays a MIB file as a graphic MIB map tree diagram that shows the MIB's structure. It allows you to analyze the MIB's content.

You can use MIBwalker to create a data table containing database entries from a group of selected variables, which is a MIBlet. See “MIBlets” on page 6-19.

MIBwalker also provides a MIB browsing application offering GET, GET NEXT, and SET capability. MIBwalker can poll a node for a MIB group or managed object and display the response on the screen or write it to the database.

MIBwalker Process Flow

Figure 6-1 shows the process for using MIBwalker to create a database table in preparation for collecting and aggregating data. The process consists of obtaining the MIB file for the vendor device (node), preparing the MIB for MIBwalker by applying typographic and syntax rules and validating object types, selecting variables for the raw table, and then creating the database table.

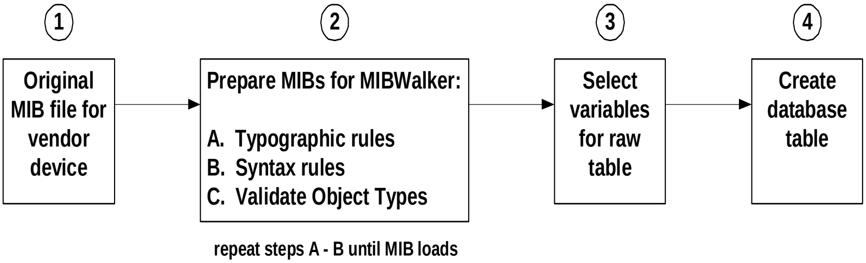

1
Original MIB file for vendor device
2
Prepare MIBs for MIBWalker:
A. Typographic rules
B. Syntax rules
C. Validate Object Types
repeat steps A - B until MIB loads
3
Select variables for raw table
4
Create database table

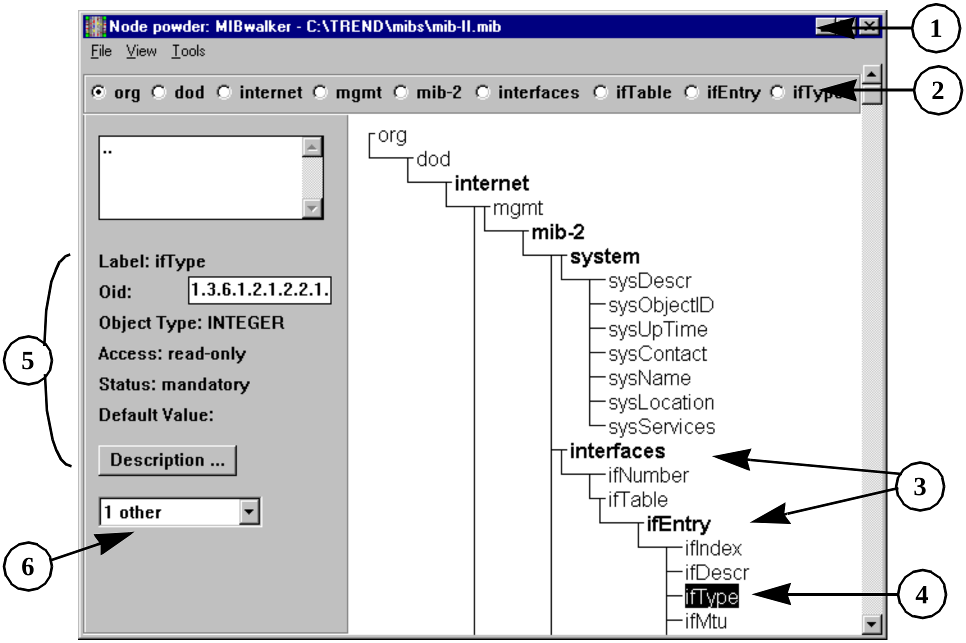
Figure 6-1: MIBwalker Process Flow

Main Display and Menu System

MIBwalker is accessed from the main TREND window. To launch MIBwalker, do the following:

1. Select **Tools** from the Menu Bar.
2. Select **MIBwalker** from the drop-down menu.

Figure 6-2 shows MIBwalker's MIB map in a tree diagram with the following items:



Label: ifType

Oid: 1.3.6.1.2.1.2.2.1.

Object Type: INTEGER

Access: read-only

Status: mandatory

Default Value:

1 other

Figure 6-2: MIBwalker Main Window

1. The header bar identifies the active node (powder), and the complete pathname of the loaded MIB file (C:\TREND\mibs\mib-II.mib).
2. The MIB path shows the path through the MIB tree to the item selected in the MIB map.
3. MIB Group names appear in bold.
4. The currently selected managed object is highlighted, which is ifType in this example.
5. The definition for the currently selected object, ifType, contains many fields. See the descriptions of the eligible fields below.
6. The enumerated list pull down menu contains the set of valid values for the selected object, ifType. This list only appears if it exists for the selected object.

The left side of the MIBwalker window displays information (item 5) that defines the currently selected managed object (item 4). To see the definition of a particular managed object, click it in the MIB map. If you click the name of a group such as mib-II, the managed object information area is blank. Eligible fields displayed for each managed object are:

Label The name of the selected managed object.

Object ID (Oid) The path through the MIB tree to the selected variable represented as a string of numbers separated by periods. The last value in the variable's object identifier is the selected variable’s position in its table.

Object Type The SNMP data type (syntax) assigned to the selected variable. The Object Type specifies the database storage definition. The aggregation process references the object type to determine the method for processing. See Table 6-1.

Valid types are: INTEGER
OCTET STRING
OBJECT ID
GAUGE
COUNTER
COUNTER32
COUNTER64
TIME TICKS
OTHER

Access The means by which a variable may be accessed.

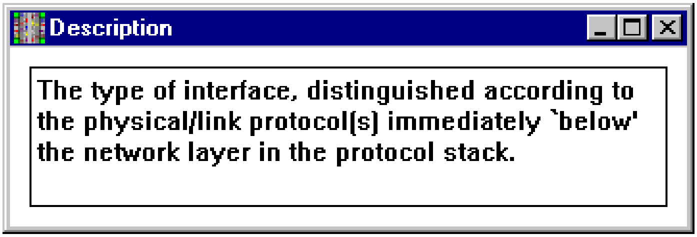
Valid access modes are: read-only
read-write
not-accessible

Status The implementation support required for the variable.

Valid access modes are: mandatory
current
optional
depreciated
obsolete

Default Value The default value for the variable, if one is specified in the MIB.

Description A pop-up window containing the description for the selected variable. Click on the Description button to display; Figure 6-3 shows the result. Note that you can stretch the window if the description is truncated.



The type of interface, distinguished according to the physical/link protocol(s) immediately `below' the network layer in the protocol stack.

Figure 6-3: Description Pop-Up Window from MIBwalker

Enumerated List A pull-down list that contains the set of valid values for the selected object. The values are usually numeric and the MIB gives the meaning for each value. This list appears only when it exists for the selected object.

The processing method for the following object types (Table 6-1) varies even though they have the same database storage type. See "Data Aggregation and Manipulation" on page 9-1 for more information regarding the rollup of different data types.

Table 6-1: SNMP Object Types Defined

SNMP Object Type	**Database Storage Type**
INTEGER	float
OCTET STRING	varchar
OBJECT ID	varchar
GAUGE	float
TIME TICKS	float

Table 6-1: SNMP Object Types Defined

SNMP Object Type	Database Storage Type
COUNTER	float
COUNTER64	float
OTHER	N/A

Note: Make sure to validate the Object Type of each variable prior to creating your MIBlet. The MIB must be modified to reflect the proper Object Type to achieve the desired processing method. See “Validating Object Types” on page 6-14.

Command Menu Option Summary

File

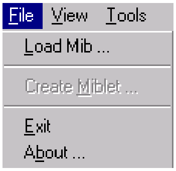


Figure 6-4: File Selection Window

The **Load Mib** option allows you to load a different MIB into MIBwalker. The name of the currently loaded MIB file appears in the header bar. See “Loading a MIB” on page 6-10.

The **Create MIBlet** option allows you to create a raw collection table in the database comprised of selected managed objects from a MIB. See “MIBlets” on page 6-19.

The **Exit** command quits the MIBwalker tool.

The **About** option provides version information for TREND. See “About TREND” on page 1-19.

View

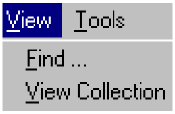


Figure 6-5: View Selection Window

The **Find** option is used to locate an object in the MIB map, provided you know all or part of the object's name.

The **View Collection** option brings up the Collect Data window. See “Collect Data” on page 7-1.

Tools

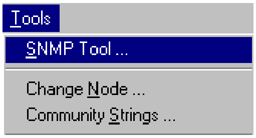


Figure 6-6: Tools Selection Window

The **SNMP Tool** polls the current node for the current values of selected MIB objects and displays the results on the screen. See “Polling Data to Screen” on page 6-31.

The **Change Node** option allows you to change the currently polled node. MIBwalker uses the current node as the target when polling for current MIB values. See “Changing the Current Node” on page 6-29.

The **Community Strings** option allows you to temporarily override community strings and specify new values to be used in MIBwalker. MIBwalker reads the community strings for the current node from the database. See “Changing Community Strings” on page 6-30.

Navigating the MIB Map

Most MIBs are too large to display all at once in the main window. MIBwalker provides several methods of walking through the structure of the currently loaded MIB. The simplest method for seeing parts of the MIB map not currently visible is to use the scroll bar to the right of the map.

Using Object Path Radio Buttons

The row of buttons immediately below the menu bar displays the path to the currently selected object in the MIB map. The following example shows the buttons that display to the `enterprises` level of the MIB tree.

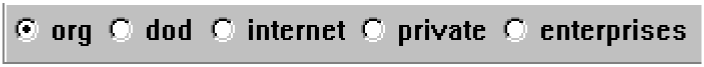
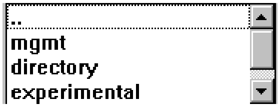


Figure 6-7: MIB Object Path Radio Buttons

Click any of these buttons to jump to the object named in the button. This is a useful way to work your way back up the MIB's tree structure. The MIB map will start with the chosen object.

Using the Next Level Box

The scrolling box located to the left of the MIB map, and just below the object path radio buttons, displays the names of the objects in the next level of the MIB tree below the currently selected object. The following example shows the scrolling box that displays for mib-II with the `internet` group selected.



..
mgmt
directory
experimental

Figure 6-8: Listing of Objects in Current Level

Click the top level “..” to move up one level of the MIB tree. Click one of the object names below the top level to navigate down the MIB tree.

Finding an Object in the MIB Map

If you know all or part of the name of an object you want to locate in the MIB map, use the Find window to go to it. Bring up the Find window by selecting the **Find** option under the **View** menu:

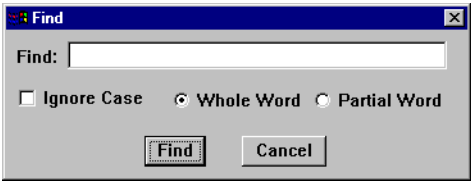


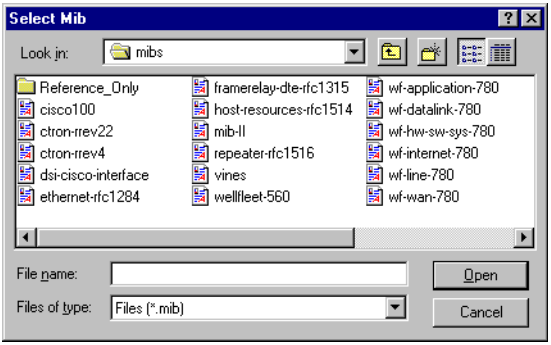
Figure 6-9: Find Window

Type in as much of the object's name as you want or know in the **Find** box. Click the **Ignore Case** check box to ignore case differences between your entry and actual object names when searching for the object. This can be useful because SNMP object names often have embedded capital letters. Indicate how much of what you typed in has to match an object name for **Find** to consider them a match by clicking on either the **Whole Word** or **Partial Word** box. If you choose **Partial Word,** Find compares your entry to any portion of the object name; however, it is case sensitive unless you select the **Ignore Case** option. Click the **Find** button to begin or continue the search. Click **Cancel** to close the window.

Loading a MIB

MIBwalker shows you the name of the currently loaded MIB file in the window's header bar. To load another MIB, do the following:

1. Select **File>Load Mib**. The following screen appears showing the DPIPE_HOME/mibs directory by default:



Look in: mibs

Files of type: Files (*.mib)

Figure 6-10: Select MIB Window

2. Select a MIB. The MIB name appears in the **File Name** field. Click **Open**.

   The newly loaded MIB name appears in the window's header bar and the MIB appears in the MIB map.

Preparing MIBs for MIBwalker

MIBwalker uses standard Structure of Management Information (SMI)-compliant SNMP MIBs. Raw MIB files must be modified to comply with typographic and syntax rules set by the MIB parser. Object types must be validated as well, and the MIB must be properly named. Once the MIB meets these requirements it can be loaded successfully.

Typographic Rules

Lines in a MIB file must be terminated by a carriage return (CR) and a line feed (LF). Lines cannot be terminated by a CR alone.

Syntax Rules

The following MIB syntax rules apply:

Top of MIB Tree

You must include the top of the ISO Tree down to start of the MIB in the file. Make sure the following lines are included:

```
internet     OBJECT IDENTIFIER    ::=    { iso org(3)
                                         dod(6) 1 }
directory    OBJECT IDENTIFIER    ::=    { internet 1 }
mgmt         OBJECT IDENTIFIER    ::=    { internet 2 }
```

Below this point, the appropriate part of the tree that reaches the start of the provided MIB must be added. For a vendor MIB, you most likely need:

```
private       OBJECT IDENTIFIER   ::=    { internet 4 }

enterprises   OBJECT IDENTIFIER   ::=    { private 1 }
```

Many standard MIBs defined in RFCs are under the MIB-II tree. For these MIBs, add the following after the line defining mgmt.

```
mib-2   OBJECT IDENTIFIER    ::=    { mgmt 1 }
```

Description Length

DESCRIPTION clauses containing more than 3000 characters are truncated to the first 3000 characters.

Object Name Length

The maximum length of a MIB object name is 63 characters. Object names of 64 characters and above are truncated to 63 characters. Exceeding the standard length however, does not affect your ability to collect or report on the field since the field is polled by its object identifier which is unchanged. The field name is a just a label.

STATUS Field Values

The legal operands for the STATUS field are:

```
mandatory
current
optional
deprecated
obsolete
```

The following value is not allowed and should be replaced with a legal one:

```
experimental
```

Convert SNMP v2 Statements

- Comment out the MODULE-IDENTITY section.
- Comment out the MODULE-COMPLIANCE section.
- Remove the label TEXTUAL-CONVENTION from statements where it occurs.
- Replace MAX-ACCESS with ACCESS.
- Replace BITS with OCTET STRING.
- Replace TimeStamp with TimeTicks.
- RowStatus is actually a TEXTUAL-CONVENTION. Replace it with:

  ```
  SYNTAX     INTEGER {
                active(1),
                notInService(2),
                notReady(3),
                createAndGo(4),
                createAndWait(5),
                destroy(6)
               }
  ```

  Other TEXTUAL-CONVENTIONs are defined in RFC1903.
- Replace multi-part SIZE arguments with a range that includes the smallest and largest arguments.

  For example: SIZE (0 | 8 | 20) is replaced by SIZE (0..20)

Validating Object Types

It has been observed that MIBs sometimes use inappropriate object types. The most common examples of this are when fields that function as a Counter or a Gauge are defined as INTEGER (Note that the field type labels are case sensitive.). These must be corrected if the data collected from those fields is to be aggregated correctly.

The DESCRIPTION field can give you a clue about the way the data in a field is used. Fields that should be defined as Counters monitor how many of something occur over time. Their DESCRIPTION field often contains the word *count* or *total number*, as in “Total number of octets since the device booted”. For example, the following field should be defined as a Counter:

```
sysLinkCellsTx
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"Count of ATM cells transmitted on the link"
::= { sysLinkEntry 10 }
```

The correction follows:

```
sysLinkCellsTx
SYNTAX Counter
ACCESS read-only
STATUS mandatory
DESCRIPTION
"Count of ATM cells transmitted on the link"
::= { sysLinkEntry 10 }
```

Fields that function as a Gauge usually indicate how much or how many of something there are at any given point in time. Their DESCRIPTION field often contains the words *in use*, *used*, or *available*. For example, this field should be defined as a Gauge:

```
sysCurrentBuffers
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of system buffers currently in use."
::= { sysLinkEntry 22 }
```

The correction follows:

```
sysCurrentBuffers
SYNTAX Gauge
ACCESS read-only
STATUS mandatory
DESCRIPTION
"The number of system buffers currently in use."
::= { sysLinkEntry 22 }
```

These data types should be corrected before data is collected from the MIB so the database tables use the appropriate field types.

Note: If the changed variable appears in a SEQUENCE OF list, change the type in the sequence list too.

MIBlet Names

Data table names in TREND have two names, Object Name and SQL Name. The Object Name is the name the user sees when selecting a table through the TREND GUI, for example when defining data collection or creating a report. The SQL Name is the actual name of the table in the database.

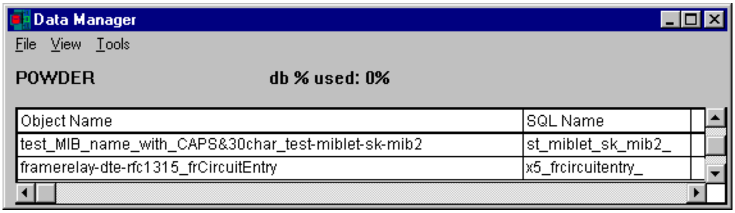
The Object name is derived by combining the name of the MIB file with the name of the MIBlet, joined by an underscore (_). The file name of the MIB file can be anything allowed by the operating system. DeskTalk recommends that you do not use dots or periods in the file name other than the .mib extension, since the mib file named used in the Object name is truncated at the first dot or period in the MIB file name.

The corresponding SQL name is created according to the following rules:

- The string is converted to all lower case characters.
- Hyphens are replaced with underscores.
- If the string is more than 17 characters long, it is truncated at the 17th character from the right.
- If the resulting first character is not a letter, it is changed to an *x*.
- An underscore is appended to the resulting name.
- If the resulting SQL name is unique, it becomes the name used in the database.
- If the resulting SQL name is not unique, the trailing underscore is changed to a zero (0). If it is still not unique, the zero increases by one until the name is unique, and that becomes the table name.

If you are using a MIB file named **framerelay-dte-rfc1315.mib** and you create a MIBlet named **frCircuitEntry** the resulting Object name is framerelay-dte-rfc1315_frCircuitEntry; the resulting SQL name is x5_frcircuitentry_ (Figure 6-11).

If you are using a MIB file named **test_MIB_name_with_CAPS&30char.mib** and you create a MIBlet named **test-miblet-sk-mib2** the resulting Object name is test_MIB_name_with_CAPS&30char_test-miblet-sk-mib2; the resulting SQL name is st_miblet_sk_mib2_ (Figure 6-11).



POWDER db % used: 0%

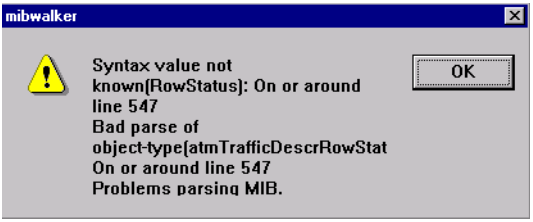
Object Name	SQL Name
test_MIB_name_with_CAPS&30char_test-miblet-sk-mib2	st_miblet_sk_mib2_
framerelay-dte-rfc1315_frCircuitEntry	x5_frcircuitentry_

Figure 6-11: MIBlet Name Examples in Data Manager

Note: If a vendor has multiple versions of their MIB, the version number should be preserved in the MIB file name before the .mib extension. To accomplish this, append the version number to the file name with a character other than a period. For example, version 1.1 of a MIB called Desktalk.mib, is identified by naming the file Desktalk-1-1.mib, Desktalk1-1.mib or Desktalk_1-1.mib. If the file is called Desktalk.1.1.mib or Desktalk.11.mib, the .1.1.mib or .11.mib component is dropped from the name of the database table. Add the version number after the name so the number is retained if the name is truncated, since it is truncated from left to right.

Troubleshooting a MIB

Current MIB parser technology used in MIBwalker requires strict adherence to typographical and syntax rules as described in “Preparing MIBs for MIBwalker” on page 6-11. If the parser rejects the MIB, a popup box is output by the parser, with one or more error messages. These messages identify an error message, the line where the rule violation is located, and a reason for the error. See Figure 6-12.



Syntax value not known(RowStatus): On or around line 547
Bad parse of object-type(atmTrafficDescrRowStat On or around line 547
Problems parsing MIB.

Figure 6-12: Example of an error message

The rejected MIB file must then be modified to correct these errors. Make the necessary changes and reload the MIB file.

Recommended Procedure

DeskTalk recommends the following procedure for troubleshooting a MIB that will not load:

1. Make a copy of the rejected MIB and rename the file.
2. Open a text editor and examine the MIB contents.
3. View the entire MIB in terms of separate vendor objects and logical blocks. The objective is to first divide the MIB into manageable and proportionate blocks, then to diagnose each block until you have isolated the area that caused the MIB to be rejected.
   a. Retain text from the beginning of the file, down to the end of Block 1.
   b. Cut all text from the beginning of Block 2 to the end of Block 4.
   c. Retain the “END” statement.
   d. Save the file. Figure 6-13 illustrates a basic MIB scheme.

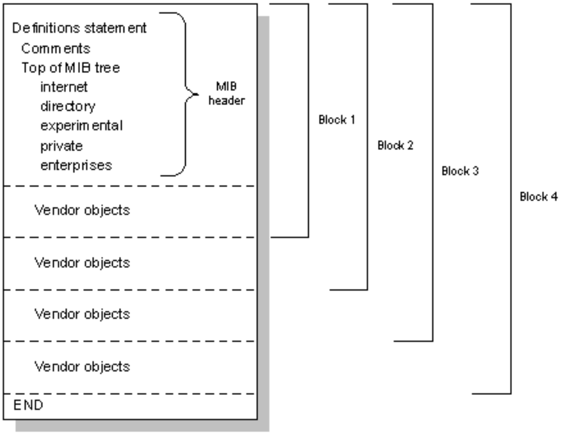


Figure 6-13: Viewing a MIB in Distinct Blocks

4. Using the TREND interface, select **File>Load Mib**. If this file loads properly, you have confirmed that this portion of the MIB is in compliance with the rules.

5. Now add the Block 2 text to the file and repeat the procedure. Do this for the remaining blocks in the file. By process of elimination, you should be able to isolate the block(s) that contain rule violations.

MIBlets

MIBlets can provide a data source for custom designed reports. A MIBlet is a composite set of MIB variables selected for data collection. MIBlets are useful in that they allow you to focus your collection on data actually needed for reports. This increases polling efficiency, speeds up data collection, and reduces the amount of database space used. More importantly, MIBlets allow you to generate a single report that includes variables from separate MIB tables.

Utilizing 64-Bit Values

SNMPv2 introduced 64 bit counters; however, SNMPv1 agents only support 32-bit counters. One common way that SNMPv1 agents return the higher precision values stored in 64-bit counters is to overlay a pair of 32-bit counters over the 64 bit field and provide the full value in two pieces.

In the following example, a 64-bit counter called `sitCounter` is broken into a high order 32-bit field called `sitCtrSplit1High` and a low order 32- bit field called `sitCtrSplit1Low`. By collecting the two 32-bit fields and reassembling them, TREND is able to store and process the full 64-bit value.

```
sitCtrSplit1High OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The high 32-bits of the 64-bit counter, sitCounter."
    ::= { sitEntry 7 }
```

```
sitCtrSplit1Low OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The high 32-bits of the 64-bit counter, sitCounter."
    ::= { sitEntry 8 }

sitCounter OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The SectionInTransit units received since the device
       booted."
    ::= { sitEntry 9 }
```

It is critical that the two halves be reassembled in the correct order. “High Part” refers to the high order, most significant 32-bit portion of the 64- bit counter. “Low Part” refers to the low order, least significant part of the 64-bit counter.

64 Bit Counter	
High Order 32 bit counter	**Low Order 32 bit counter**

Creating a MIBlet

MIBwalker allows you to select a specific set of MIB managed objects from one or more groups so the selected objects can be polled together in a single table. This is known as a MIBlet. This section provides a description of the steps for creating a MIBlet.

Note: You do not need to create a MIBlet to run any of the existing ReportPacks provided by TREND. Necessary MIBlets are created automatically.

CAUTION

When creating a MIBlet, you must be sure to select only objects that are commonly indexed. MIB objects in the same table are, by definition, commonly indexed. MIB objects from different tables may be commonly indexed. For example, some enterprise MIBs contain tables indexed by interface number just like MIB-II ifEntry is indexed.

Creating a MIBlet containing objects that are indexed differently results in invalid data in your database.

Note: The maximum width for a table created from a MIBlet is 2,000 characters. If you create a table that exceeds this width, a diagnostic message appears when you run TRENDit.

Perform the following steps to create a MIBlet:

1. Decide which variables (MIB objects) you wish to include in your MIBlet and select those objects by clicking each one with the right mouse button.

Figure 6-14 shows that a box appears around each selected object:

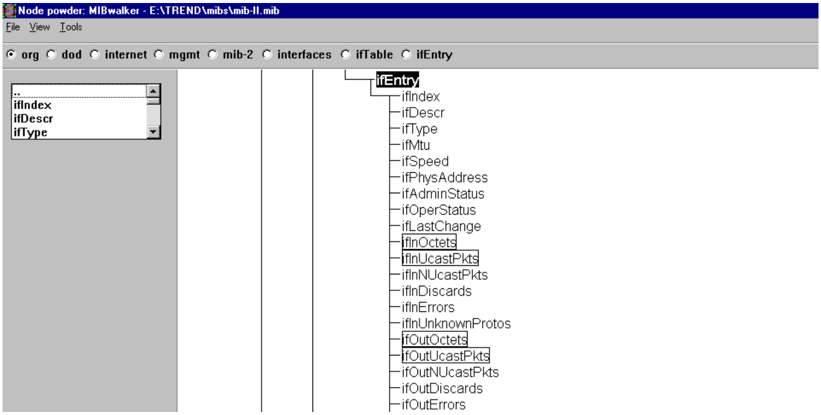
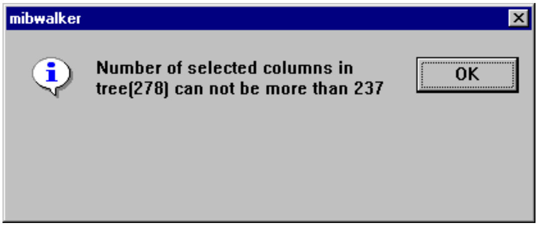


Figure 6-14: MIBwalker, with Variables of the ifEntry Table Selected

If you select a variable by mistake, deselect it by clicking the right mouse button again. To deselect all the variables, click the left mouse button.

You may select an entire group of variables by clicking the group name that appears in bold. However, if you select some variables in the group and then select the group, you are actually selecting the remainder of variables in the group.

If you select more than 237 variables for a MIBlet, the following message appears.



Number of selected columns in tree[278] can not be more than 237

Select fewer variables to create the MIBlet.

2. From the **File** pull-down menu, select **Create MIBlet.**

   This option is only accessible when there are MIB objects selected for inclusion in a MIBlet.

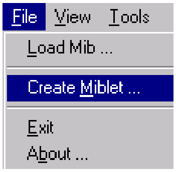
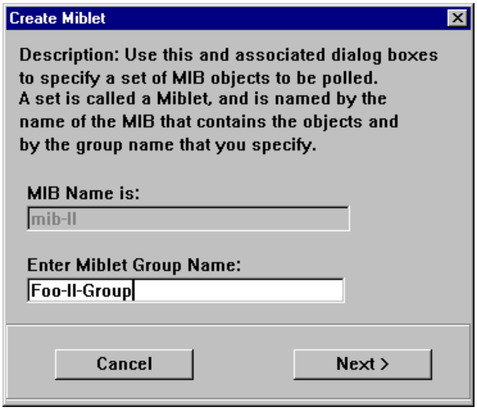


Figure 6-15: File Menu, With Create MIBlet Selected

The Create MIBlet window appears:


Create Miblet

Description: Use this and associated dialog boxes to specify a set of MIB objects to be polled. A set is called a Miblet, and is named by the name of the MIB that contains the objects and by the group name that you specify.

MIB Name is: mib-II

Enter Miblet Group Name: Foo-II-Group

Figure 6-16: Create MIBlet Window

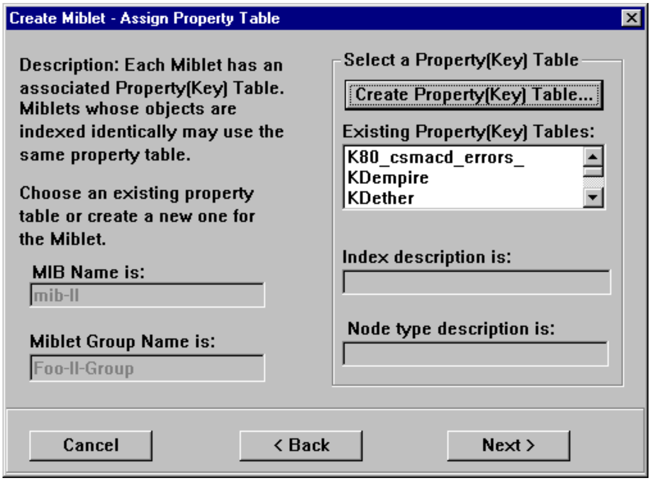
The name of your MIB file, up to the first dot, appears in the **MIB Name Is:** field.

3. Enter a group name in the **Enter Miblet Group Name:** field.

   This name is appended to the MIB name with an underscore between the names to create the Object name that appears in Data Manager.

4. Click **Next>**.

The Create MIBlet - Assign Property Table window appears.



Create Miblet - Assign Property Table

Description: Each Miblet has an associated Property(Key) Table. Miblets whose objects are indexed identically may use the same property table.

Choose an existing property table or create a new one for the Miblet.

MIB Name is: mib-II

Miblet Group Name is: Foo-II-Group

Select a Property(Key) Table

Existing Property(Key) Tables:

- K80_csmacd_errors_
- KDempire
- KDether

Index description is:

Node type description is:

Figure 6-17: Create MIBlet - Assign Property Table Window

The elements polled for data in the MIBlet may be defined either by their own property table, or an existing property table. The Create MIBlet procedure gives you the choice of either creating a new property (key) table or identifying an existing one for the MIBlet. The latter should be used whenever possible since it saves space in the database, and allows the properties of elements to be provisioned by the existing table. MIBlets that are collected for a common element type must use the same property table. If you choose to associate your MIBlet with an existing property table, it is **very important** that you choose the correct table. Remember that if you choose any other property table, you risk invalidating the data in your new table and any other derived from it.

Note: If you create a MIBlet, and are unsure which existing property table is appropriate, you should choose to create a new property table. This guarantees that your MIBlet does not interfere with other data tables.

Note: The property table you select must be indexed in the same way as the objects in your MIBlet.

5. Select a Property (Key) table.

   a. To select an existing property table:

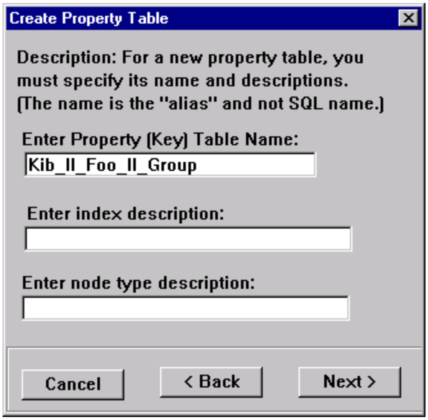
      In the **Existing Property (Key) Tables:** field, select the property table you wish to be associated with your MIBlet.

      Upon selection, the index description appears in the **Index description is:** field of the MIBlet's data table, and the attributed node type is displayed in the **Node type description is:** field.

   b. To create a new property table for the MIBlet:

      Click on **Create Property (Key) Table** button.

      The Create Property Table window appears.



Create Property Table

Description: For a new property table, you must specify its name and descriptions. (The name is the "alias" and not SQL name.)

Enter Property (Key) Table Name: Kib_II_Foo_II_Group

Enter index description:

Enter node type description:

Figure 6-18: Create Property Table Window

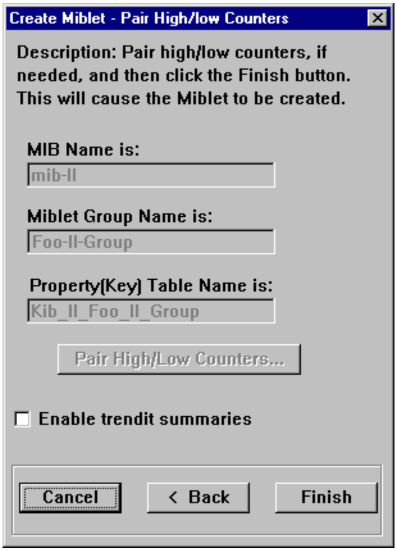
This dialog box (Figure 6-18) defines the property table that identifies your MIBlet.

You may enter an alias name, such as Router_Port_Properties, in the **Enter Property (Key) Table Name:** field. This name will appear in the **Existing Property (Key) Tables:** field on the Create MIBlet - Assign Property Table window.

Enter a description of the instance identifier, such as port, into the **Enter index description:** field.

Enter a node type description, such as router, into the **Enter node type description:** field.

6. Click **Next** to continue.
7. The Create Miblet-Pair High/low Counters window appears (Figure 6-19).


Create Miblet - Pair High/low Counters

Description: Pair high/low counters, if needed, and then click the Finish button. This will cause the Miblet to be created.

MIB Name is: mib-II

Miblet Group Name is: Foo-II-Group

Property(Key) Table Name is: Kib_II_Foo_II_Group

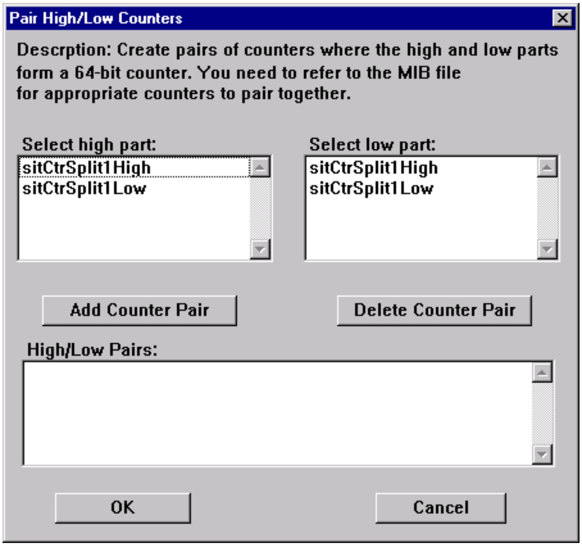
Figure 6-19: Create Miblet-Pair High/low Counters Window

MIBwalker automatically creates the raw and rate MIBlet tables with all other required SQL objects to support raw-to-delta processing.

8. To create the legacy summary tables from TRENDit, select the **Enable trendit summaries** check box.

9. If you do have 64-bit counters, then click **Pair High/Low Counters** button. This button is accessible when the selected MIB objects are 32-bit counters.

   The Pair High/Low Counters window appears.



Descrption: Create pairs of counters where the high and low parts form a 64-bit counter. You need to refer to the MIB file for appropriate counters to pair together.

Select high part:
sitCtrSplit1High
sitCtrSplit1Low

Select low part:
sitCtrSplit1High
sitCtrSplit1Low

Figure 6-20: Pair High/Low Counters Window

   a. Select one entry from **Select High Part** to act as the high counter, and one entry from **Select Low Part** to designate the low counter. The selection appears as the following example:

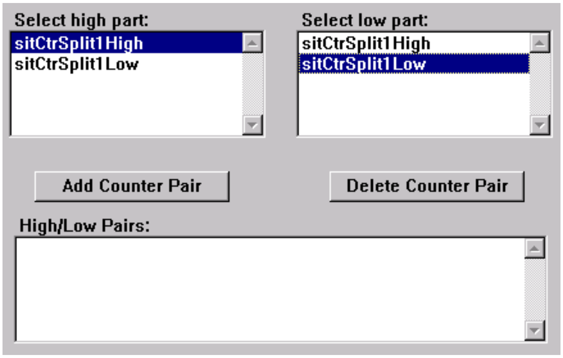
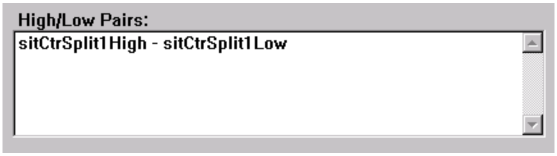


Figure 6-21: Counter Pairing Selection

b. Click on **Add Counter Pair**. The pairing appears in the **High/Low Pairs**, and is removed from the selection blocks. See Figure 6-22.

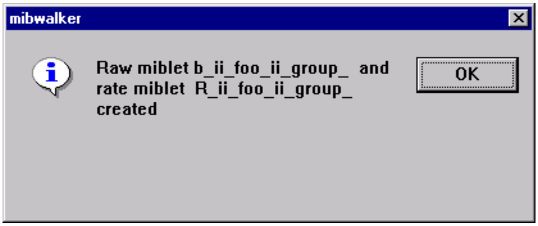

High/Low Pairs:

sitCtrSplit1High - sitCtrSplit1Low

Figure 6-22: View of Applied Pairing

10. Click **Finish**.

11. MIBwalker displays the following message, and creates the raw and rate MIBlet tables with all other required SQL objects to support raw-to-delta processing.



Raw miblet b_ii_foo_ii_group_ and rate miblet R_ii_foo_ii_group_ created

Figure 6-23: Window Verifying Creation of Tables

Once the MIBlet tables have been created you can use TREND's Collect Data tool to begin collecting data for it. See "Collect Data" on page 7-1 for a detailed description of Collect Data.

Utilities

Changing the Current Node

MIBwalker uses the current node, named in the header bar, as the target in the SNMP Tool. You can change the current node to a different device. Select the **Change Node** option in the **Tools** menu to bring up the Connect To window.

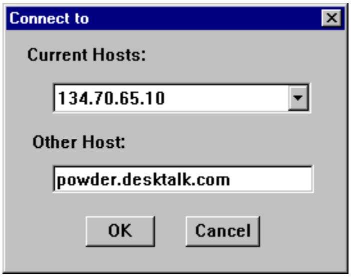


Figure 6-24: Connect To Dialog Box

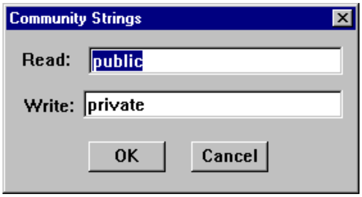
The window displays the name of the current node on the **Other Host** line. The list of nodes already known to TREND is in the **Current Hosts:** box. To change the current node, type in a new node name or the node's IP address on the **Other Host** line, or select a node from the list of **Current Hosts** by clicking on the down arrow next to the **Current Hosts** box.

- Click the **OK** button to make this the new current node.
- Click the **Cancel** button to quit out of the window without changing the current node.

The name of the current node is displayed in the MIBwalker Main window header bar.

Changing Community Strings

MIBwalker uses the current community strings in the SNMP Tool. If the current node is known to TREND, MIBwalker uses the community strings that are configured for the node in TREND. New nodes use the default. The default community strings are *public* for read, and *private* for write. You can specify new values to be used in MIBwalker. Select the **Community Strings** option in the **Tools** menu to bring up the Community Strings window:



Read: public
Write: private

Figure 6-25: Community Strings Window

The current read and write community strings are displayed in their respective boxes in the window. Type in new values for either or both strings.

- Click the **OK** button to have the new values take effect.
- Click the **Cancel** button to quit out of the window without changing the community string values.

MIBwalker reads the community strings for the current node from the database each time you change nodes, so the values you enter here are only in effect until you change nodes. Changing the community strings for the current node in this window does not update their values for the node in the database. Periodic polling of the node by `mw_collect` uses the community strings in the database, not the ones specified here.

Polling Data to Screen

The SNMP Tool allows you to Get and display data for MIB values on the current node. To use the SNMP tool:

1. Select a MIB object by clicking the right mouse button on its name in the MIB map. This highlights the object. In the following example **mib-II.mib** is loaded, with the system variable **ifEntry** selected.

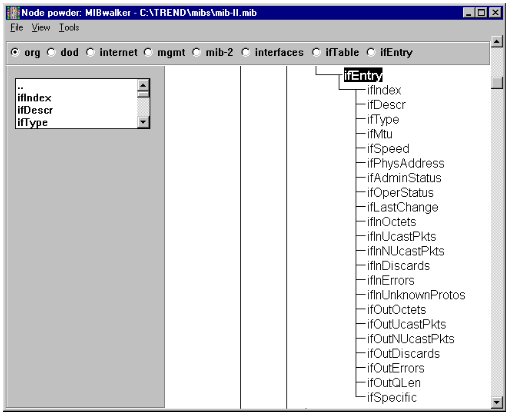


Figure 6-26: MIBwalker, with Variables of the ifEntry Table Selected

2. Select the **SNMP Tool** option from the **Tools** menu to bring up the SNMP Tool window.

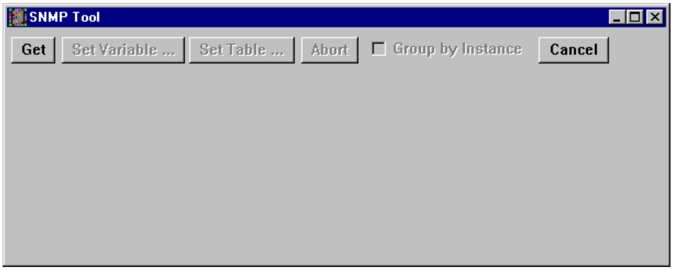
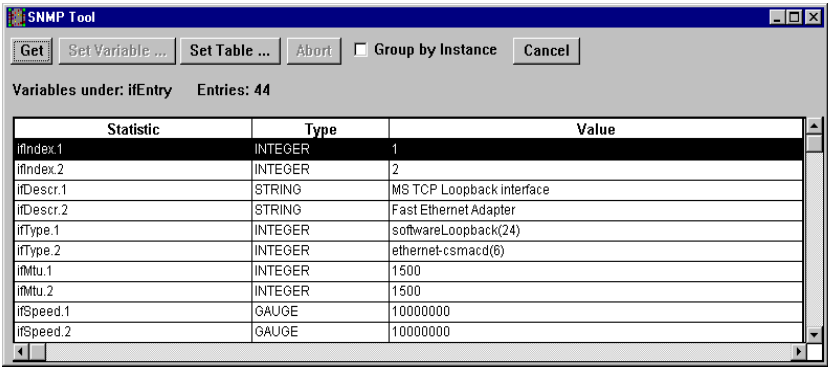


Figure 6-27: SNMP Tool Window

3. Click the **Get** button in the SNMP Tool window to initiate a GetNextRequest tree walk of the MIB tree for the group or object selected. Figure 6-28 below shows a partial listing of entries received for a `GET` on the ifEntry variable.



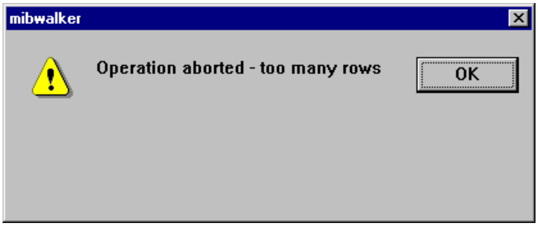
Variables under: ifEntry   Entries: 44

Statistic	Type	Value
ifIndex.1	INTEGER	1
ifIndex.2	INTEGER	2
ifDescr.1	STRING	MS TCP Loopback interface
ifDescr.2	STRING	Fast Ethernet Adapter
ifType.1	INTEGER	softwareLoopback(24)
ifType.2	INTEGER	ethernet-csmacd(6)
ifMtu.1	INTEGER	1500
ifMtu.2	INTEGER	1500
ifSpeed.1	GAUGE	10000000
ifSpeed.2	GAUGE	10000000

Figure 6-28: SNMP Tool with ifEntry Variable Displayed

If you select a specific variable, only that variable is polled, but you get the value of that variable for all instances of the table it is in. If you select a table, MIBwalker polls for every variable in every instance of that table, including the contents of any tables under the selected table.

A counter on the left side of the window below the Get button increases in increments of 50 as data is received up to 2000 entries. If the Get operation produces more than 2000 entries for your selection, the following screen appears.



Operation aborted - too many rows

Figure 6-29: Too Many Rows Error Message

The 2000 rows collected are still shown in the SNMP tool window.

Once the GetNext completes, the number of variables returned is displayed in the message:

```
Variables under: <object> Entries: <number>
```

where:

`<object>` is the name of the object selected to be collected in the MIB map

`<number>` is the number of values returned

If you do not want to wait for polling to finish, you can abort the Get by clicking on the `Abort` button. Polling stops, and any values already received are displayed.

If a table has multiple instances, SNMP Tool displays the data in the table grouped by variable. That is, all the values for the first field are displayed, then all the values of the next field, and so on. If you want to display the data such that all the variables for one instance are grouped together followed by all the variables for the next instance, click the **Group by Instance** button. The fields are displayed in the same order in which they are defined in the MIB table.

Something not always obvious when collecting data from nodes on the network is whether or not each device actually returns a value for every field in the requested table. With the SNMP Tool, it is easy to see whether this is the case. Poll the current node for the table. Do not group by instance. Scroll through the returned values and see if any fields are missing.

SNMP Sets

You can set the values of writable MIB variables through the SNMP Tool window, either picking out a single variable or updating any writable variables in a table simultaneously. The current node is the target of the set.

Setting A Single Variable

1. Select the variable name in the MIB map that you want to set or the table containing that variable.
2. Bring up the SNMP Tool window (Figure 6-27) if it is not up already.
3. Click the **Get** button in the SNMP Tool window.

   If you selected the specific variable in the MIB map, only that variable appears in the SNMP Tool window. In the event there are multiple instances of the variable, all instances appear. If the variable is writable, the **Set Variable** button is highlighted when the variable value is returned.

   If you selected and got an entire table, select the instance of the variable you want to set. If the variable is writable, the **Set Variable** button now is highlighted.
4. Click the **Set Variable** button. This brings up the Set Variable window (Figure 6-30). Note that the SNMP Tool window will be frozen while the Set Variable window is up.

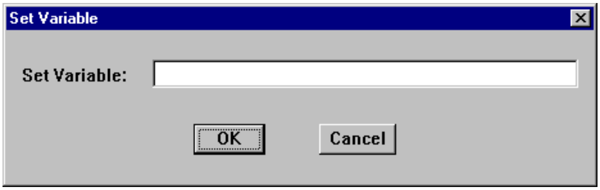


Figure 6-30: Set Variable Window

5. Enter the new value for the variable on the Set Variable line.
6. Click the **Apply** button to set the variable. If the set is successful you will see a box with the following message on your screen:

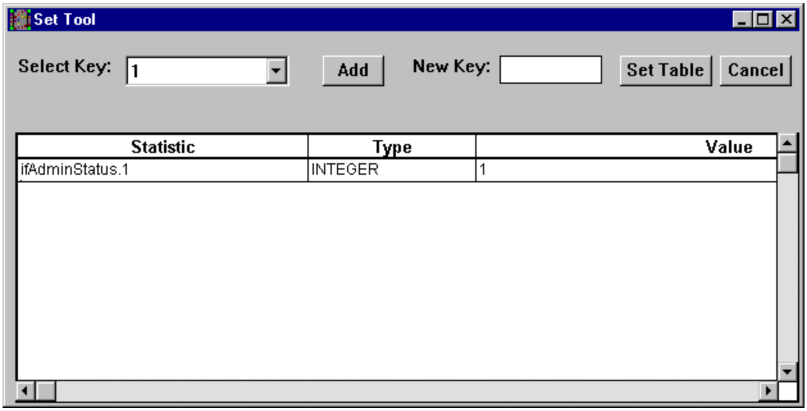
```
set successful
```

7. Click the **OK** button in the box to continue.
8. Click the **Cancel** button to quit out of the window.

Setting Multiple Variables in a Table

1. Select the table name in the MIB map that contains the variable you want to set.
2. Bring up the SNMP Tool window if it is not up already.
3. Click the **Get** button in the SNMP Tool window. The **Set Table** button is highlighted.

   Note that the Set Table button is highlighted any time a table containing writable variables is polled in SNMP Tool.
4. Click the **Set Table** button. This brings up the Set Tool window.



Select Key: 1

New Key:

Statistic	Type	Value
ifAdminStatus.1	INTEGER	1

Figure 6-31: Set Tool Window

The SNMP Tool window will be frozen while the Set Table window is up. The window lists all the writable variables in the current table, their data types and current values:

5. Select the instance ID for the instance whose variable(s) you want to set by clicking on the down arrow of the **Select Key** box then selecting the desired value. The variables and current values for the selected instance are displayed. By default, the first instance and its associated field values are shown.

6. Enter new values in the value column for the fields you want to set. Click in another field or press the Return key to enter the value. You can change keys and set values on fields with different keys as part of this same step.
7. When all fields are entered the way you want, click the **Set Table** button to apply the new values to the target node. If the set is successful you will see a box with the following message on your screen:

```
set successful
```

8. Click the **OK** button in the box to continue.
9. Get the table again to verify the values are correct.
10. Click the **Cancel** button to quit out of the window.

Adding a New Key

You can add a new instance to the current table with the Set Tool window if the current node allows it for the current table.

1. Enter the value for the new key on the **New Key** line, then click the **Add** button. If the operation is successful you will see a box with the following message on your screen:

```
set successful
```

2. Click the **OK** button in the box to continue.

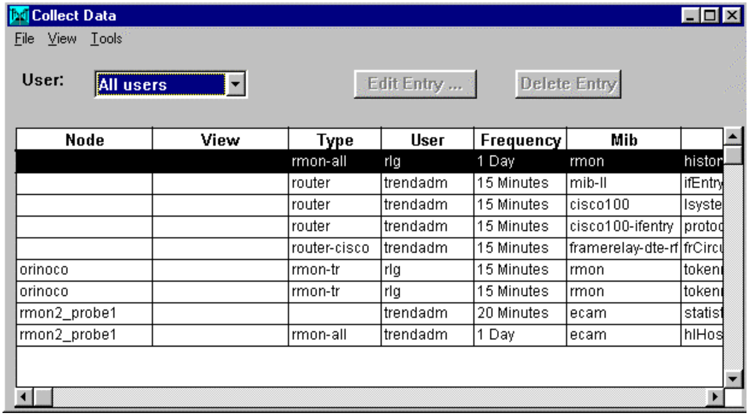
7 Collect Data

The Collect Data application's main functions are to define an enterprise polling policy for regular collection of management data and to administer the storage of data in a relational database. It is easy to use because requests are set up through a point-and-click GUI, and it is powerful because a single collection request can be pointed at a group of nodes, or at a type of node, and all appropriate devices are polled automatically. The list of devices is dynamic; as nodes are added to or removed from the group, polled devices change accordingly. The user can modify or delete current polling definitions as necessary.

Note: To add MIB variables not currently in your TREND database, see "MIBwalker" on page 6-1**.**

Collect Data

The application is invoked by clicking on the Collect Data button in the main TREND window:



User: All users

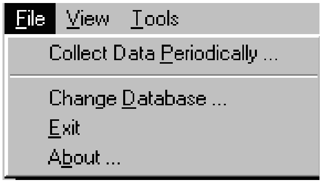
Node	View	Type	User	Frequency	Mib	
		rmon-all	rlg	1 Day	rmon	histor
		router	trendadm	15 Minutes	mib-II	ifEntry
		router	trendadm	15 Minutes	cisco100	lsyste
		router	trendadm	15 Minutes	cisco100-ifentry	protoc
		router-cisco	trendadm	15 Minutes	framerelay-dte-rf	frCircu
orinoco		rmon-tr	rlg	15 Minutes	rmon	tokenr
orinoco		rmon-tr	rlg	15 Minutes	rmon	tokenr
rmon2_probe1			trendadm	20 Minutes	ecam	statist
rmon2_probe1		rmon-all	trendadm	1 Day	ecam	hlHos

Figure 7-1: Collect Data Window

Menu Command Option Summary

The Menu Bar of the Collect Data window has three pull down menus: File, View, and Tools.

The File Menu



The File menu contains four options: **Collect Data Periodically, Change Database, Exit, and About. Exit** and **About** are standard commands, and are described in "Commonly Used Commands" on page 1-15.

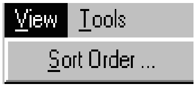
Collect Data Periodically

Use the **Collect Data Periodically** option to define new data collection parameters for a selected table (see "Specifying How Often to Poll for the Data" on page 7-7).

Change Database

Select the Change Database option to view and collect data from tables in alternate databases (see "Changing Databases" on page 7-9).

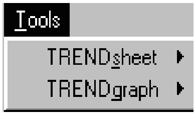
The View Menu



The View menu currently has only one option: **Sort Order.**

Sort Order is used to select the sort order of the polling information, from the list of items available (see "Reordering the List of Collection Requests" on page 7-11).

The Tools Menu



The Tools menu contains two options: **TRENDsheet** and **TRENDgraph.**

Create a simple tabular or graphical display of data in a database table by invoking **TRENDsheet** or **TRENDgraph** from the Tools menu.

Note: For RMON data, only Control Tables can be viewed. TRENDsheet and TRENDgraph displays of RMON raw and roll-up database tables are not available.

Setting Up New Polling Requests

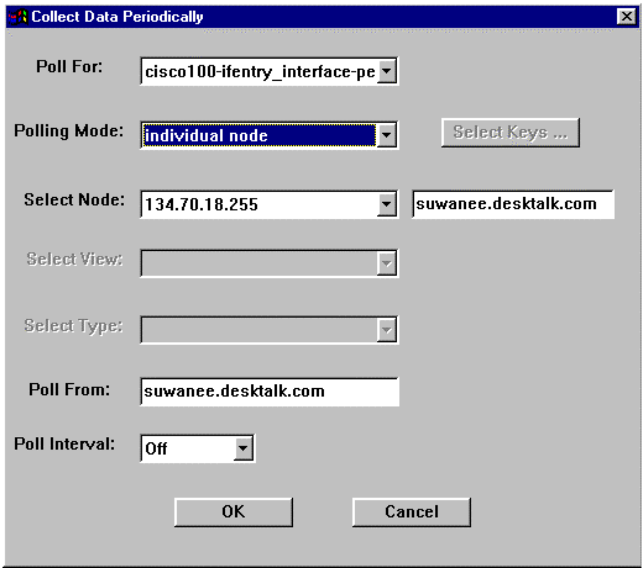
Collect Data provides a mechanism for defining and managing the set of polling requests that comprise an SNMP and RMON collection policy. TREND's polling agents collect data from nodes on your network according to the request parameters defined. This information includes:

- What data to collect
- Which nodes to poll
- How often to poll
- Which system does the polling
- Where to store collected data

Specifying What Data to Collect

TREND collects data on variables (i.e., MIB objects) and groups of individual variables called MIBlets.

To define the polling parameters for the selected group of variables or table, click the **Collect Data Periodically** option under the File menu. This will bring up the Collect Data Periodically window:



Poll For: cisco100-ifentry_interface-pe

Polling Mode: individual node

Select Node: 134.70.18.255 suwanee.desktalk.com

Poll From: suwanee.desktalk.com

Poll Interval: Off

Figure 7-2: Collect Data Periodically Window

The **Poll For** box in the window displays the names of the variables or tables that can be defined. Click on the down arrow next to the box to see a list of variables and tables available for selection.

Specifying Which Nodes to Poll

The **Polling Mode** box lets you define the polling target grouping for this collection request. Membership in the list of target nodes is determined at each polling cycle, so if nodes are added or deleted, the change will automatically affect which nodes are polled for this data. Click on the down arrow next to the box to see a list of the nodes and select the one you want. Alternatively, type in a node name (if Community String values are public and private).

The possible values are:

individual node

Polls only the node named on the **Select Node** line. This is the node named in the header bar of the window. You can change the default node name using the **Node List** box. Click on the down arrow next to the box to see a list of the nodes available, and select the one you want.

individual node by key

TREND allows you to conduct directed instance polling using previously assigned keys. Use the Select Keys button to view and select keys which define the data to be collected for a given node.

all nodes of same type

Each node in your network is some type of device. It can be useful to poll all similar nodes for common data. Select this option to have TREND poll all nodes of the selected type for this collection request.

If you select this mode, the Select View box is grayed out and the **Node** lines in the window are ignored. Select the desired type by clicking on the Type box’s down arrow then clicking on the desired type from the displayed list. The value you select will appear on the **Select Type** line.

all nodes in view

The nodes in your network can be grouped into views. A view may be based on a common location, function, or other criteria that indicate a set of devices that you want to share a common polling policy. Select this option to have TREND poll all nodes in the view you select for this collection request.

The user must belong to a group to use the following modes.

If you select this mode, the **Select Type** box is grayed out, and the **Select Node** and **Select View** lines in the window are ignored. Select the desired view by clicking on the View box's down arrow then clicking on the desired view from the displayed list. The value you select will appear on the **Select View** line.

all nodes of same type in view

This mode lets you focus a target list even further, by causing only nodes of the specified type, which are also members of the specified view, to be polled for the selected MIB table.

If you select this mode, you must next select one value from the **Select View** box and one value from the **Select Type** box. The values you select will appear on the **Select View** and **Select Type** lines.

Select the desired **Polling Mode** from the box, then select the values for the mode as described above. The values displayed on the **Node, View,** and **Type** lines are selected from the corresponding boxes as described above, not as input lines.

Specifying How Often to Poll for the Data

TREND polls for requested data at regular intervals as specified in the collection request definition. To select a polling interval for the current request, click on the down arrow next to the **Poll Interval** box, then click on the desired interval. If you select Off, this request will be ignored.

Specifying Which System Should Poll for the Data

It can be more efficient to poll from a system closer to the intended targets than from the one on which the request was defined. This approach reduces network overhead and distributes the polling load. By default, the system on which polling is defined also does the polling. You can override this by entering a different client name on the **Poll From** line. The node name entered here must be configured to run TREND polling agents.

Collections Involving Multiple Database TREND Installations

From the user's perspective, TREND's use of multiple databases for data collection is completely transparent. In addition to the database containing the polling policy definition, each polling action based on a view and/or a type may involve up to two other databases. Use of other databases is based on the name of the user who defined the polling action. The user's name can be seen in the User column of the Collect Data window. Each user is a member of a group. Each group has an associated topology database and a data database. Node, view, and type definitions are stored in the topology database. Different logical topology databases associated with different user groups may share a common physical database. Once collected, data is stored in the data database.

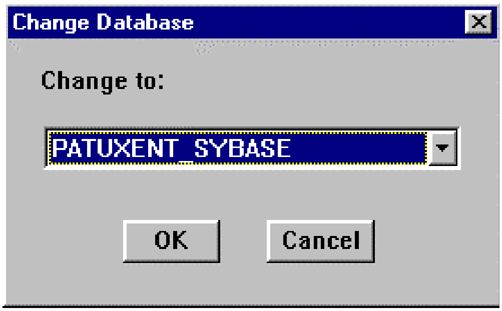
The SQL Server, which stores collected data, is set in the data_db field of the tl_groups table in the database. The group, of which a user is a member, is set in the group_name field in the tl_users table. The user, whose group_name and associated data_db value is used to determine where to store the data, is the user who defined the data request. If a user is not a member of a group, the data_db entry for the group *default* is used.

Saving the Collection Request Definition

Once the collection request is defined the way you want it, save the definition by clicking OK. The definition will appear in the Collect Data window. Click **Cancel** to discard the current definition and exit the window.

Changing Databases

You can access a different SQL Server by clicking on the Change Database option under the **File** menu to bring up the Change Database window:



Change to: PATUXENT_SYBASE

Figure 7-3: Change Database Window

Note: A report may not display the same information for both databases.

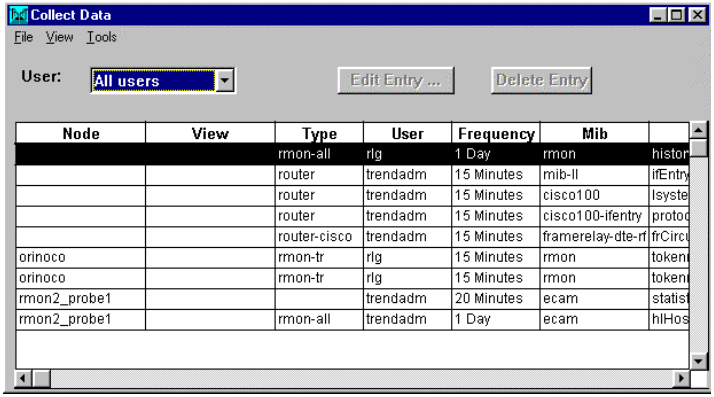
The name of your current database is displayed in the **Change to** box. To change to a different server, click on the down arrow in the **Change to** box to see a list of the Sybase SQL Servers known to this machine. The values presented here are read from the $SYBASE/interfaces file:

- Select the name of the server you want to access in the report.

- Click OK to change to the selected database. Click Cancel to quit out of the window without changing servers.

Managing Polling Requests

It is important to keep track of your data collection requests and to be able to modify or delete them. The Collect Data window displays a list of all data collection request definitions stored in the database:



User: All users

Node	View	Type	User	Frequency	Mib	
		rmon-all	rlg	1 Day	rmon	histor
		router	trendadm	15 Minutes	mib-II	ifEntry
		router	trendadm	15 Minutes	cisco100	lsyste
		router	trendadm	15 Minutes	cisco100-ifentry	protoc
		router-cisco	trendadm	15 Minutes	framerelay-dte-rf	frCircu
orinoco		rmon-tr	rlg	15 Minutes	rmon	token
orinoco		rmon-tr	rlg	15 Minutes	rmon	token
rmon2_probe1			trendadm	20 Minutes	ecam	statist
rmon2_probe1		rmon-all	trendadm	1 Day	ecam	hlHos

Figure 7-4: Collect Data Window

The table in the window lists all pertinent information about each request definition as setup in the Collect Data Periodically window. The following columns are in the table:

Node/View/Type The combination of values in the first three columns is determined by the polling mode. If an individual node is being polled, the Node and View columns contain values (the Type value is not used here). If the request is polling by Type, only the Type column contains a value, and so on.

User The name of the user who defined the request.

Frequency How often TREND will poll for the requested data.

Mib The Category or SNMP MIB that contains the table being collected.

Group The Group or MIB table being collected.

Poll From The name of the system that actually polls for the data.

Which Request Definitions Are Displayed

The list of requests displayed in the window is controlled by the User box. Only requests with a User column value matching the value in the User box are displayed. By default, the User value is the name of the user running TREND. To change the value in the box, click on the down arrow next to the User box and select the desired value from the list of all users presented.

Reordering the List of Collection Requests

You can change the sort order of the displayed requests. Click the **Sort Order** option in the **View** menu to bring up the Order Requests window:

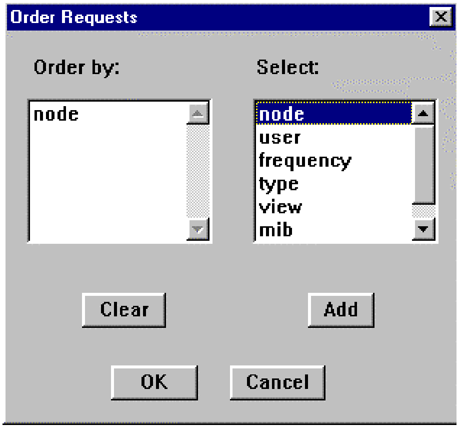


Figure 7-5: Order Requests Window

The current sort order is shown in the **Order by** box. By default, the requests are sorted by **node** name. To change this, first click on the **Clear** button to remove the current sort order. Next, click on a column name in the **Select** box that you want used as the new sort criteria. Activate the selected value by clicking on the **Add** button (you can add more than one field to the **Order by** list). The order in which you add the fields determines the sort order of your results. Collect Data does a nested sort beginning with the first field listed in the **Order by** box, then the next field within the first, and so on.

- When the Order by list is as desired, put it into effect by clicking on the OK button.
- Click on the Cancel button to exit the window. If you cancel without applying a new sort order, the order will not change.

Modifying Request Definitions

Existing requests can be modified by the user who created them, as shown in the **User** column, or by the user *trendadm*. To modify a request, click on the request’s line in the table to highlight that request’s row, then click on the **Edit Entry** button (alternatively you can double-click on the line). This will bring up the Collect Data Periodically window. Modify the polling parameters as described in “Setting Up New Polling Requests” on page 7-4. The new parameters will be displayed in the table.

Deleting Request Definitions

Existing requests can be deleted by the user who created them, as shown in the **User** column, or by the user *trendadm*. To delete a request, click on the request's line in the table. This will highlight that request's row. Next click on the **Delete** button. The row will be removed from the table.

Exiting the Collect Data Window

To quit out of the Collect Data window, select **Exit** from the **File** Menu.

Some Considerations When Organizing Polling

TREND makes it very easy to setup data polling that will collect a great deal of information from a large number of nodes, even from just a few polling requests. It is worth taking a moment to plan your polling strategy.

Polling Groups

TREND's ability to poll user defined groups of devices from a single data collection request is one of the application's most powerful features. When defining device groups, there are various strategies that can be used to maximize efficiency and simplify administration. Not all of these suggestions apply to every site. Use the ones that suit your needs:

Group By Function In general, a particular type of device plays a particular role in the network. This may be obvious, but it needs to be taken advantage of by actually creating a polling group for nodes that provide that function. For example, create a group that contains all routers, all file servers, or all hubs. While TREND supports polling by device type, different types may be assigned to the same kind of device from different vendors or even to different models from the same vendor. Functional groups overcome that distinction.

Group By Information Available You may choose to create node groups according to a particular type of information available from all the nodes in the group.

Group By Location In a large network, it may make sense to subdivide groups based on the devices’ physical locations so polling can easily be distributed to the polling system closest to the target nodes. Grouping by location makes it simple to setup polling of group Location-A by Poller-A, and of Location-B by Poller-B.

Group By Vendor While all devices of a particular type can be polled as group by specifying **Poll by Type** in the request definition, it may be the case that different types are assigned to the same kind of device from different vendors. For example, the type used for Vendor A’s router may be different than the type used for Vendor B’s router. Vendor functional groups overcome that distinction.

Group By Priority This can be viewed as a way to subset larger groups. For example, in a large network, there may be dozens or hundreds of routers. You may want to focus on the core routers for polling purposes. Create a *Core Routers* group, which is a subset of the *Routers* group discussed above.

Remember that the polling processes, ee_collect, mw_collect and rmon_collect, read the membership list for the group being polled each time it starts a polling cycle. Changes made to the group are automatically reflected in the list of nodes actually polled for the data.

Note: Views and types are defined separately for each user group. Polling must be defined by a user in that group for polling to occur. Any user can only belong to one group.

Polling Frequency

The type of data collected in a polling request, and the reason it is collected, directly affect your decision regarding how often to poll for the data. For example, system configuration data that rarely changes, or disk utilization data that changes gradually, may only need to be collected once a day. Dynamic performance statistics, with characteristics that vary with the time of day, will need to be collected many times each day to give an accurate picture. The number of minutes between polls also depends on how much the data is likely to change in the interval.

Other factors affecting polling frequency decisions include; whether you are likely to miss anything if the polls are farther apart, the load placed on both the polled and polling systems, and on the network by the data collection (especially if target nodes are at remote sites), and on the amount of database space you have for storing the data. Remember that changing polling frequency from 10 to 5 minutes doubles the volume of data collected. Typically performance statistics are collected every 10 or 15 minutes, while very large tables such as RMON host and matrix are collected hourly.

Additionally, keep in mind that TREND is not intended for reviewing real-time, current statistics to solve an immediate problem. That is the role of dashboard or real time applications which collect data at a high frequency rate, generally measured in seconds, display the most recent data, and discard it when new data arrives or soon afterwards. Such real time applications are focused on troubleshooting, with a few key statistics collected from a few devices. TREND polling is aimed at providing a baseline to support the troubleshooting activity and at identifying patterns generally occurring over hours, days and weeks.

Viewing SNMP Data

TREND lets you display the data you've collected on your screen. This can be data recently polled from the target node, or it can be data previously stored in the database in raw or aggregated form. Data is viewed via TREND's spreadsheet style reporting application, TRENDsheet, or line graph style report application, TRENDgraph. Normally, the TRENDbuild application is used to select what data to include in a report.

You can quickly create a simple report of the data in a database table by invoking **TRENDsheet** or **TRENDgraph** from the **Tools** Menu in the Collect Data window.

Select the collection request definition that polls for the table you want to look at. Select the type of display by clicking on the arrow in the TRENDsheet or TRENDgraph button. This will display a pop-up menu of the available types of data. Click on the type value you desire.

The following options are available:

View Raw Data	Displays raw sample data.
View Rate Data	Displays inter-sample delta values for counters, and original values for all other fields.
View Hourly Data	Displays hourly summary information.
View Daily Data	Displays daily summary information.
View Weekly Data	Displays weekly summary information.

These default reports include all statistics in the selected table without aliases or expressions. As with all TREND reports, a query file is created. These reports can therefore be manipulated like any other graph or table report (see "TRENDsheet" on page 11-1 and "TRENDgraph" on page 12-1). If you requested a TRENDgraph, the Select Data Set window will be displayed in order to define the line styles for the fields being plotted.

Command Line Options

The TREND polling processes **ee_collect**, **mw_collect** and **rmon_collect** are launched from the command line. See "Man Pages" on page A-1 for a more detailed description of these applications.

Incomplete Data Collection or Aggregation

Data collection (the running of ee_collect, mw_collect, or rmon_collect) and aggregation (the running of TRENDit, TRENDsum, and other aggregation utilities) place bcp (i.e., bulk copy) and other temporary files in the following default directories:

For UNIX:	/tmp
For Windows NT:	c:\tmp

For Windows NT, the temporary files are placed in c:\tmp by default. You can change this default location for Window NT by assigning the desired path to the variable %DPIPE_TMP%, as follows:

1. Sign on to the local domain as Administrator.
2. Select Start, Settings, Control Panel, System, Environment.

3. Assign the desired path to %DPIPE_TMP%. For example:

```
DPIPE_TMP = f:\tmp
```

If the /tmp directory in UNIX or the %DPIPE_TMP% (or default c:\tmp) directory in Windows NT does not exist, the TREND data collection and aggregation operations are *not* performed successfully. No error message is issued.

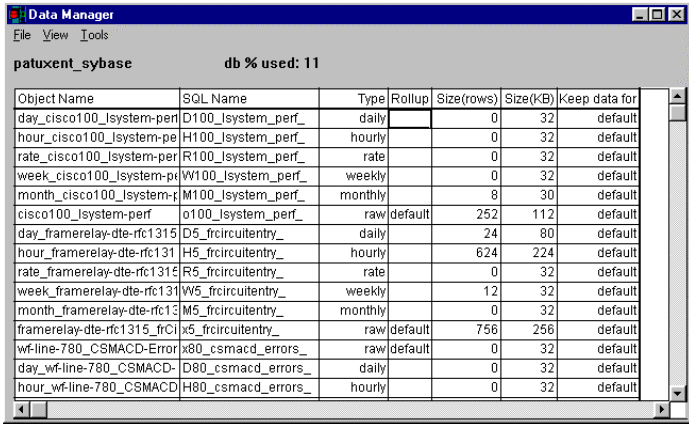
If you are having problems with incomplete data collection or aggregation operations, ensure that the /tmp directory in UNIX or %DPIPE_TMP% (or c:\tmp) directory in Windows NT exists. If not, create it.

8 Data Manager

Data Manager is an administrative tool for monitoring the use and growth of a TREND database. The TREND user needs a way to monitor the database, which is continually changing due to automatic table creation, population, rollup and aging. Data Manager provides an easy to read tabular display of information about every data table in your database(s), plus a central location for controlling data rollup and aging. Any TREND user can view information about the database via Data Manager. Only **trendadm**, however, can use Data Manager to change defaults, modify aging and rollup for individual tables, and delete and truncate tables.

Main Display and Menu System

Invoke the Data Manager screen from the main TREND window.



patuxent_sybase db % used: 11

Object Name	SQL Name	Type	Rollup	Size(rows)	Size(KB)	Keep data for
day_cisco100_lsystem-perf	D100_lsystem_perf_	daily		0	32	default
hour_cisco100_lsystem-pe	H100_lsystem_perf_	hourly		0	32	default
rate_cisco100_lsystem-per	R100_lsystem_perf_	rate		0	32	default
week_cisco100_lsystem-pe	W100_lsystem_perf_	weekly		0	32	default
month_cisco100_lsystem-p	M100_lsystem_perf_	monthly		8	30	default
cisco100_lsystem-perf	o100_lsystem_perf_	raw	default	252	112	default
day_framerelay-dte-rfc1315	D5_frcircuitentry_	daily		24	80	default
hour_framerelay-dte-rfc131	H5_frcircuitentry_	hourly		624	224	default
rate_framerelay-dte-rfc1315	R5_frcircuitentry_	rate		0	32	default
week_framerelay-dte-rfc131	W5_frcircuitentry_	weekly		12	32	default
month_framerelay-dte-rfc13	M5_frcircuitentry_	monthly		0	32	default
framerelay-dte-rfc1315_frCi	x5_frcircuitentry_	raw	default	756	256	default
wf-line-780_CSMACD-Error	x80_csmacd_errors_	raw	default	0	32	default
day_wf-line-780_CSMACD-	D80_csmacd_errors_	daily		0	32	default
hour_wf-line-780_CSMACD	H80_csmacd_errors_	hourly		0	32	default

Figure 8-1: Data Manager Main Window

Database Information

The Database Summary Information area, located between the menu bar and the tabular display, provides the following information:

- **<SQL Server Name>**

  TREND can read and store data on any SQL server in the user’s network, so it is important to be able to monitor all servers from a single location. This box indicates which server’s tables are currently being displayed.

- **db% used:**

  This tells the user how much of the space available in the TREND database on the SQL server currently being monitored has been used. The database is full at 96%. Data collection will cease at 90% until the size of the database has been reduced.

Tabular Display

The tabular portion of the main Data Manager window includes the following:

- Object Name

  The user visible name of a table in the database. It is composed of the combined names of the MIB and its subordinate table (or schema and group) that is the source of the data in the database table.

- SQL Name

  The name by which the table is known to the SQL Server and the name which must be used in any SQL queries.

- Type

  The kind of data stored in this table. Possible values are data (raw sample data), rate (inter-sample delta values for counters, original values for all other fields), hourly, daily, weekly, monthly, and internal (an internal TREND table that cannot be truncated or deleted).

- Rollup

  Shows and sets whether or not the data in this raw data table will be aggregated into rate, hourly, daily and weekly summaries. Possible values are yes (aggregate), no (don't aggregate), and default (aggregation determined by the value set for Roll Up Data Automatically in the Default Storage Time window. (See "Changing System-wide Rollup and Aging Defaults" on page 2-14.) Only the user **trendadm** can change this value.

- Size (Rows)

  The number of rows in the table.

- Size (KB)

  The size of the table in kilobytes.

- Keep Data for

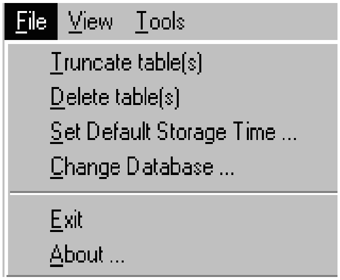
  Shows and sets the number of days the data will be kept in the table. Data is discarded when it is **aging + 1** days old. Possible values are forever (never delete), # Days (keep the data in this table for # days), and **default** (determined by the value set for Keep <*type*> Data in the Default Storage Time window, where <*type*> matches the Type column value for this table. (See “Changing System-wide Rollup and Aging Defaults” on page 2-14.)

This field is blank for tables which are not aged (e.g. internal tables). Only the user **trendadm** can change this value.

Command Menu Option Summary

The menus available from the menu bar of the main Data Manager window are described below.

File



The **Truncate table(s)** command removes the contents of a table without deleting the table. Truncation does not write the removed rows to the transaction log, so it cannot be recovered.

Only the user **trendadm** can use this function (see “Truncating Tables” on page 2-16).

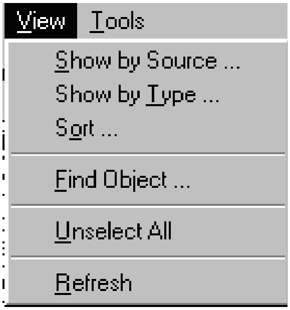
Delete table(s) drops the table and its contents from the database. The deleted table is written to the transaction log. If the table is too large to be written to the transaction log (it will need space twice its size in transaction log for the delete to succeed), use the **Truncate** option to empty the table before deleting it.

Only the user **trendadm** can use this function (see “Deleting Tables” on page 2-17).

Set Default Storage Time brings up a window that shows the values that will be used to control rollup and aging for tables that do not have specific values set (see “Changing System-wide Rollup and Aging Defaults” on page 2-14).

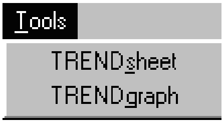
The **Change Database** command provides access to data tables in other databases. Tables are read out of the $SYBASE/interfaces file.

View



- Select **Show by Source** to have Data Manager only display values for tables from the selected source.
- Select **Show by Type** to have Data Manager only display values for tables of the selected type.
- Select **Sort** to sort the table rows in the specified order.
- Select **Find Object** to search Data Manager for a particular table by whole word or partial word.
- Select **Unselect All** to deselect table rows highlighted.
- Select **Refresh** to update the Data Manager display with current information on the state of the database. The information is not updated on the fly; you must use the Refresh command to view the latest changes.

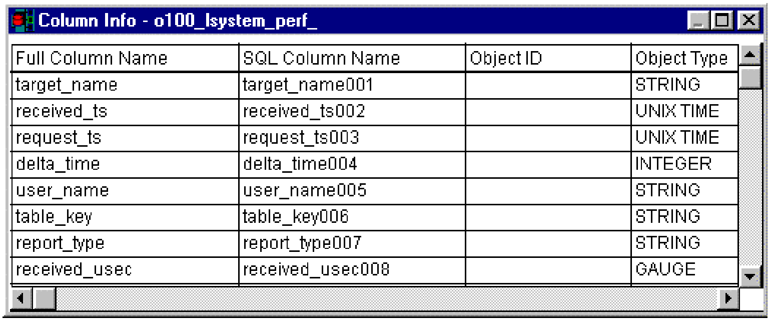
Tools



View the data in a table by selecting the appropriate row, then enter **TRENDgraph** or **TRENDsheet** directly from Data Manager via the Tools menu.

Obtaining Table Profile Information

By double-clicking on a row in the table, you can view information that describes each column in that table: full column name, SQL column name, object ID, and object type.



Full Column Name	SQL Column Name	Object ID	Object Type
target_name	target_name001		STRING
received_ts	received_ts002		UNIX TIME
request_ts	request_ts003		UNIX TIME
delta_time	delta_time004		INTEGER
user_name	user_name005		STRING
table_key	table_key006		STRING
report_type	report_type007		STRING
received_usec	received_usec008		GAUGE

Figure 8-2: An Example of the Column Info Window

Showing Particular Tables

By default, Data Manager shows information about all data tables and most internal tables. If you only want to see a particular group of tables, you can specify a group either by source or by type. To select a class of table, do the following:

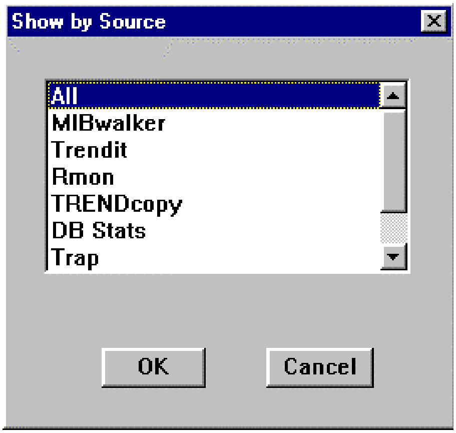
1. Click the **View** button in the Menu Bar.
2. Select the category by which you want to subset the tables. Your choices are:

   Show by Source

   Show by Type

   When you select a category, a popup window will show you the choices in that category.

   If you selected Show by Source, the popup looks like this:



Show by Source

All
MIBwalker
Trendit
Rmon
TRENDcopy
DB Stats
Trap

Figure 8-3: Show by Source Window

If you selected **Show by Type**, the pop-up looks like this:

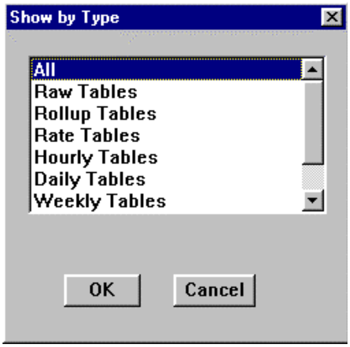


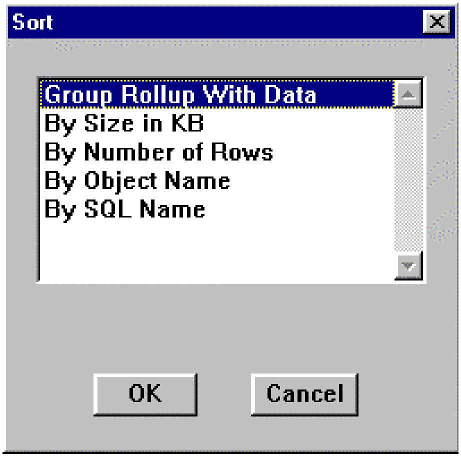
Figure 8-4: Show by Type Window

3. Click the desired value in the popup window.
4. Click **OK** to have your choice take effect. Click **Cancel** to leave the contents of the Data Manager window unchanged.

Changing the Sort Order of the Data Manager Window

By default, Data Manager sorts the rows in the Data Manager window area by grouping a raw data table and its associated rollup tables together. You can resort the rows by following these steps:

1. Click the **View** button in the Menu Bar.
2. Select **Sort** from the **View** menu. This popup window will show you the fields which can be used to sort the rows:



Sort

Group Rollup With Data
By Size in KB
By Number of Rows
By Object Name
By SQL Name

Figure 8-5: Sort Window

3. Select the *field* on which you want to sort the rows.
4. Click **OK** to have the new sort order take effect. Click **Cancel** to leave the sort order unchanged.

Monitoring the Rate of Database Growth

The database occupies a fixed amount of pre-allocated space on the disk. Although TRENDsystem provides a number of features for controlling the growth of the data occupying this space, it is important to monitor how quickly it is being used up and what is using it up so the growth can be controlled and accommodated.

The first place to watch for database space use is the **db% used** in the **Database Summary** area. Besides keeping track of what percent of available space is left, track how quickly that space is getting used up.

Next, watch the size (in Kbytes) of your data tables. Data Manager's sort feature can make this easier. If you are using up space quickly, see which tables are largest and how fast they are growing. Control their growth by shortening their aging periods. Decide what kind of data you really need to keep for an extended period of time versus those you can discard after a day or two. If you are not reporting on raw data for example, you probably don't need to keep it longer than the day it needs to be maintained for aggregation. Since this is the most voluminous data it makes sense to manage it more carefully. In extreme cases you may need to truncate a large table to keep the database from filling up. The longer the data table, the longer it will take for the report to run.

Note: Your database is becoming full at 80% and is completely full at 96%. Adjustments are recommended if db% used exceeds 70%. Do your housekeeping periodically and delete tables you no longer need.

You can obtain a report that predicts database size by using AutoPilot to install the Database Size ReportPack. See the TREND 3.6 ReportPack Guide.

9 Data Aggregation and Manipulation

TREND polling utilities called *DataPipes* collect performance data from the devices in your enterprise. TREND DataPipes can collect raw data from Simple Network Management Protocol (SNMP), Remote Monitoring (RMON), RMON2, and other agents depending on the device. The names given to the raw data items and their data types are determined by the Management Information Base (MIB) associated with the devices being polled.

Most collected data is not readily usable in its raw form. The raw data must be processed to make it reportable in terms that provide usable performance information about the enterprise. Four TREND utilities work together to process the raw data, consolidate it, and finally display it in the reports supplied with the TREND ReportPacks:

- TRENDit performs some validity checks on the raw collected data and turns SNMP counter data into delta data. This process is called *cleaning*.
- TRENDit can also summarize the data in standard ways. More specifically, TRENDit can compute counts, averages, minimums, and maximums; and summarize an element property at the hourly, daily, weekly, and monthly level. This process is called *rollup*.

- TRENDstep creates the table on which all reports in a ReportPack are based. This table is called the *base table*. TRENDstep creates the base table for each installed ReportPack.

  To create the base table, TRENDstep examines each row of data that is input from a cleaned source for values that satisfy one or more predefined conditions. TRENDstep can conditionally process groups of rows. This capability can prevent divide-by-zero conditions, help process RMON2 data, and grade device performance.

- TRENDsum processes the data in a ReportPack's base table and groups data by the element properties and time periods, thereby creating the levels of summarization and consolidation needed for the reports. TRENDsum calculates standard statistics such as counts and averages and more complex metrics such as utilization forecasts and rolling baselines used to compare element performance.
- TRENDrank ranks performance according to defined categories such as %utilization, congestion, or Grade of Service (GOS), so you can keep track of things like the top ten best (or worst) performers. TRENDrank also tracks the change in an element's ranking, so you can identify stable and volatile resources easily.

Figure 9-1 summarizes this processing.

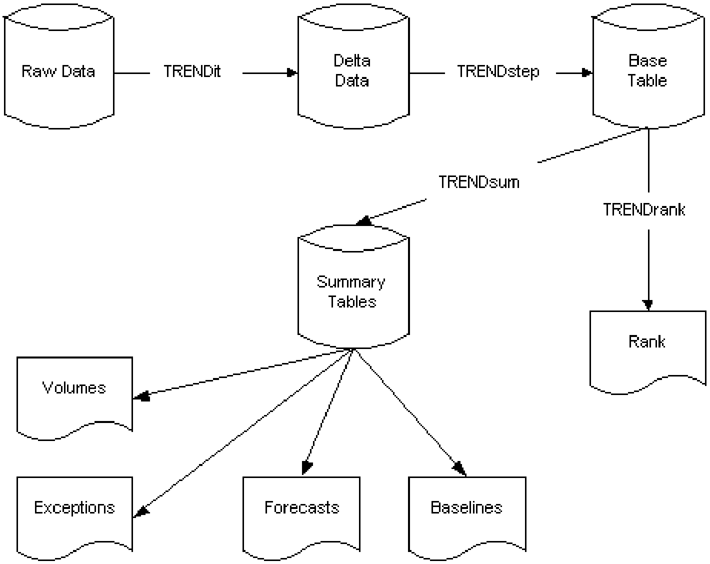


Figure 9-1: Data Aggregation and Manipulation Process

The remainder of this chapter provides an overview of how these processing functions are performed and relates the processing to TREND-supplied ReportPacks.

"Man Pages" on page A-1 describes the syntax and detailed usage of each utility.

Raw Data Collection

You can specify the interval at which the various TREND DataPipes collect the performance data that ultimately appears in the TREND ReportPacks. For purposes of this discussion, assume that a 15-minute collection interval is defined for each DataPipe (although, in practice, the collection interval typically varies by DataPipe).

Assuming a 15-minute collection interval, TREND DataPipes attempt to collect four samples per hour, or 96 samples per day (24 hours * 4 samples per hour).

TREND DataPipes store the collected data in tables, called *raw* tables, in the TREND database. In practice, data for a given sampling period is not all collected at the same instant in time; data collection may take several seconds or minutes to complete, depending on the number of devices being polled and the volume of data that is collected.

Each row of collected data has real timestamps that indicate when the request for the data is made and when the data is received. In addition, TREND DataPipes associate a virtual timestamp (which appears in a column named ta_period) with each row in the raw table. The virtual timestamp standardizes the real timestamps by applying the same begin-time value to each row of data in the sample. For example, all data collected during the 12:15 a.m. collection period on June 22, 1998 is assigned a ta_period value of Jun 22 1998 12:15 AM.

Raw data tables, then, contain rows of data items for each sampling period. All rows with the same virtual timestamp (ta_period) belong to the same sample.

Data Cleaning

TRENDit cleans the data that has been collected in raw tables and stores the results of the cleaning process in rate tables. The cleaning process has two parts:

1. SNMP counter data is transformed into delta data.
2. Raw samples are checked to see if they are valid.

Transforming Counter Data into Delta Data

The value of an SNMP counter increases continuously from zero by 1 until the counter reaches some predefined maximum value. At that point, the counter starts over at zero. Thus, the value of an SNMP counter data type is an integer that has no meaning unless it can be related to a time interval. For example, a counter named inoctets counts the number of bits input to an interface on a router. If inoctets has the value 1,000,000 in a given sample, the value is meaningless unless you can state it as a rate such as 1 million bits per second. Only then can you calculate a useful metric such as throughput.

TRENDit enables you to do this by turning raw SNMP counter data into delta data. TRENDit (invoked with or without the -r option) performs this function by examining pairs of rows in adjacent sampling periods in the raw tables and creating *rate* tables as output. Each row in the rate output table represents delta data that is computed from a pair of rows from the raw input table, as follows.

Assume for this example that the first sample is taken at time t_1 on day 1 (which is 12:15 a.m. if 15-minute sampling intervals are used). At time t_2 (that is, at 12:30 a.m.), the second sample is taken; at time t_3 (12:45 a.m.), the third sample is taken; and so on.

The value v_1 for a counter data item in period t_1 is then subtracted from the value v_2 for the counter data item in period t_2. The difference (which is called a *delta* value) is stored in the rate table row associated with period t_1. For all other (noncounter) data types, the value v_2 collected at period t_2 is stored in the rate table row associated with period t_1. (Value v_1 for noncounter data items is lost.)

The time, in seconds, between the two sampling periods (t_2 - t_1), is stored in a column called delta_time. Delta_time is computed by subtracting the SysUpTime of the first sample from the SysUpTime of the second sample, dividing the result by 100, and then truncating the remainder. For the following diagrams, if a 15-minute sampling interval is used, the delta_time between all rows is approximately 900 (15 minute period * 60 seconds per minute). However, the actual delta_time could be 894, 901, 905, or any number close to 900 depending on the calculation.

The following diagram illustrates this processing.

Raw Table

ta_period	Counter Item	Non-Counter Item
t_1	v_1	v_1
t_2	v_2	v_2

>

Rate Table

ta_period	Counter Item	Non-Counter Item	Delta_Time
t_1	$c_1=v_2-v_1$	v_2	900

Likewise, the sample for period t_3 is compared to the sample for t_2. For counter data types, the value v_3- v_2 is stored in row t_2. For noncounter data, the value v_3 collected at period t_3 is stored in row t_2. The delta_time for row t_2 is again 900.

The following diagram summarizes this processing over the course of a day:

Raw Table

ta_period	Counter Item	Non-Counter Item
t_1	v_1	v_1
t_2	v_2	v_2
t_3	v_3	v_3
t_4	v_4	v_4
...	...	...
t_{96}	v_{96}	v_{96}

>

Rate Table

ta_period	Counter Item	Non-Counter Item	Delta_Time
t_1	$c_1 =$ v_2-v_1	v_2	900
t_2	$c_2 =$ v_3-v_2	v_3	900
t_3	$c_3 =$ v_4-v_3	v_4	900
...	...	...	...
t_{95}	c_{95}= v_{96}-v_{95}	v_{96}	900

Thus, during the cleaning process, pairs of rows from the raw data tables are compared; each row of the output rate table is expressed as delta data that represents the result of the comparison. At the end of day 1, then, the raw tables contain 96 rows of data—one row for each sampling period. However, the rate tables contain only 95 rows of data at the end of day 1, since the sampling taken at period t_1 is lost.

For the second day, the rate output tables contain all 96 rows, because data from the last sample taken for day 1 (at t_{96}) is compared with the first sample taken for day 2. Thus, the first row of rate data for day 2 is not lost.

As TRENDit continues to process the new samples that are collected into raw tables, it appends new rows to the rate output tables in the fashion described above.

Validity Checking

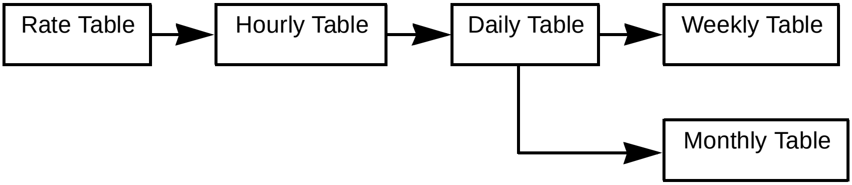
In theory, the raw data that is collected during any one sampling period is completely accurate. In practice, however, several circumstances can affect the validity of the collected data, such as:

- One or more scheduled samplings fails for some reason (e.g., the network is down).
- One or more devices returns inaccurate data (the device is malfunctioning) or no data (the device is offline).
- A counter reaches its predefined maximum value and rolls over (starts over at zero) between sampling periods.
- You manually reset some or all counters between sampling periods.
- You shut down and reboot the system between sampling periods, which resets the counters to zero.

If TRENDit senses that the sample is invalid for any of these reasons, the sample is not included in the rate output table.

Rollup

TRENDit can also perform a roll-up function on rate data (if the -r option is omitted from the trendit command line). In the roll-up function, rate data is summarized at the lowest element property (table_key) level for the time period to produce four summary tables—hourly, daily, weekly, and monthly, as follows:



During this roll-up process, the following four rows are created in each summary table for each element property (table_key):

1. A row with the minimum value for the time period (dsi_agg_type=2).
2. A row with the maximum value for the time period (dsi_agg_type=3).
3. A row with the total value for the time period (dsi_agg_type=4).
4. A row with the average value for the time period (dsi_agg_type=5).

For all data types except counter and gauge data, TRENDit stores the last value collected for the item as that item’s value in each summary row, because summaries of data types other than counters or gauges are not sensible. (For example, a device’s IP address is the same in each collection sample; averaging multiple occurrences of the IP address is a meaningless operation.) The value of the delta_time column in each summary row is the sum of the delta_time values in the input rows.

The summary rows are created as follows for counter data (which has been transformed into delta data) and gauge data:

Summary Row	**Counter (Delta) Value Computed As ...**	**Gauge Value Computed As ...**
Minimum	Minimum of the counter data values divided by their associated delta times.	Minimum of the input values.
Maximum	Maximum of the counter data values divided by their associated delta times.	Maximum of the input values.
Total	Total of all the input values	Average of all the input values. (It is not sensible to total gauge data.)
Average	Total of all the input values divided by the total delta_time.	Average of all the input values.

In the TREND database, you can tell what type of data a table contains by the first letter in the SQL name of the table. For example:

Beginning Letter	**Contents**	**SQL Table Name**
	Raw	mib_II_ifentry
R	Rate	Rib_ii_ifentry
H or **SH**	Hourly	Hib_ii_ifentry
D or **SD**	Daily (Starts on a particular day of the week)	Dib_ii_ifentry
W or **SW**	Weekly	Wib_ii_ifentry
M or **SM**	Monthly (for a calendar month)	Mib_ii_ifentry

Tables can come from any source, but must be properly typed to ensure that they are processed properly. The main Data Manager window displays this table name in the SQL Name column; the type of data in the table (raw, rate, hourly, daily, and so on) appears in the Type column. (See "Data Manager" on page 8-1.)

TRENDit provides the default set of roll-up and summarization processing functions described above for:

- The ReportPacks in TREND releases earlier than release 3.5.
- Any data table for which Yes is specified in the Rollup column.
- Any MIBlets you create on your own with MIBwalker and for which Yes is specified in the Rollup column for the item in the database. (The value given in the Rollup column is No for all tables used to generate the TREND 3.5 ReportPacks.) For a detailed discussion of MIBwalker, see "MIBwalker" on page 6-1.

TRENDsum (rather than TRENDit) creates the summary tables and calculates statistics for the ReportPacks in TREND 3.5 and later releases. See "Other Data Aggregation Functions" on page 9-15.

The Base Table

TRENDstep processes the cleaned delta data from TRENDit (or other sources) to create a template table called the *base table*. The base table contains all the columns needed to generate data for all reports in the ReportPack.

TRENDstep uses a language with if-then-else, case, and expression constructs that enables it to perform the following types of operations on the input data:

- Include source data columns in the base table without modification.
- Compute new columns from expressions using existing columns.
- Perform row-by-row conditional processing.
- Create an "other" category for RMON2 data.
- Assign grades to ranges of input data.

Row-by-Row Conditional Processing

TRENDstep can examine each row of an input table to see if it satisfies a stated condition. TRENDstep can take a rate table, a TRENDsum output table, or another TRENDstep output table as input.

TRENDstep processing is designed to:

- Provide a method for intercepting a potential divide-by-zero condition in the input data and assigning an alternative value to the result.
- Provide a way to group data into an "other" category. For example, RMON2 host or matrix data contains rows of data for many protocols. TRENDstep can combine these rows into a few columns in a single summary row.
- Assign grades to a range of input data values.

- Compute metrics such as %utilization and store them as new columns in the output table.

The Divide-By-Zero Issue

When you compute statistics, potential divide-by-zero situations can frequently arise. For example, %input utilization of an interface is typically computed as:

(((inoctets * 8) / delta_time) / ifspeed) *100

That is, inoctets (bytes) is multiplied by 8 to give a bit count, which is divided by delta_time to give bits per second. That value is divided by the interface speed (ifspeed), which is estimated bandwidth for an interface in bits per second. The result is multiplied by 100 to give a percentage.

If the interface is down for some reason when the data sample is collected, ifspeed is reported as zero, which leads to a divide-by-zero condition if utilization is computed for the interface.

TRENDstep provides a set of statements (if/else and case constructions) that you can use to conditionally process input to check for this condition. (Appendix A: *Man Pages* describes the TRENDstep language in detail.)

Another way to avoid a potential divide-by-zero problem is to use (ifspeed + 1) in the formula that computes %utilization. (The +1 increases ifspeed a negligible amount.)

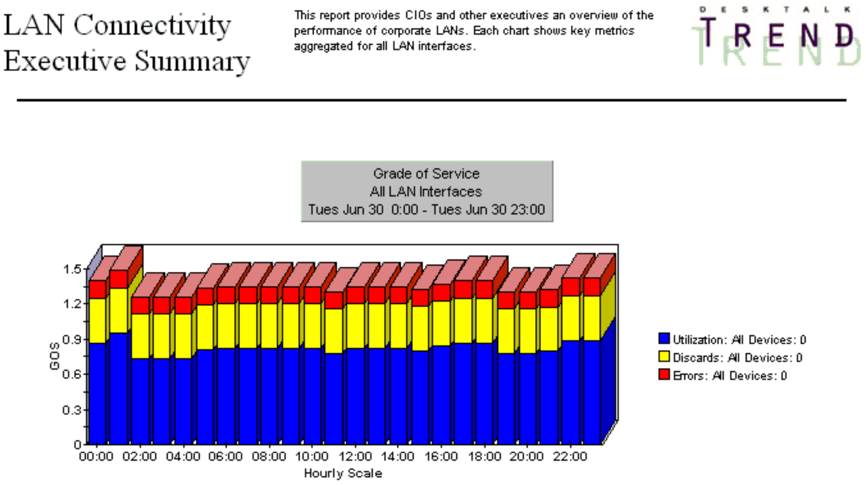
Creating an “Other” Category for RMON2 Data

RMON2 agents provide statistics on a large number of individual communication protocols that various device interfaces use. In cases where you are only interested in the statistics for a few protocols and you want to group the statistics for all other protocols into an “Other” category, you can use TRENDstep to conditionally process each column of data in the input row, create a new column called "Other" in the output table, and place the sum of the data for all other protocols in that column.

In this case, the output table contains the same number of rows as the input table, but the columns are different. The section on TRENDstep in Appendix A: *Man Pages* gives an example of this TRENDstep function.

Assigning Grades to Ranges of Input Data

The TRENDstep language also enables you to conditionally examine a selected column value in each row of input data, determine if the value falls within a particular range, and assign a grade. Grading is used to produce the Grade of Service (GOS) stacked bar chart in the Executive Summary reports supplied with the TREND ReportPacks. Figure 9-2 shows an excerpt from the LAN Connectivity Executive Summary report.



LAN Connectivity
Executive Summary

This report provides CIOs and other executives an overview of the performance of corporate LANs. Each chart shows key metrics aggregated for all LAN interfaces.

Figure 9-2: LAN Connectivity Executive Summary

Here is an excerpt from the TRENDstep script file that is used to produce this report. The excerpt illustrates how a column value from an input row is graded:

```
utilization=((ifoutoctets024+ifinoctets018)*8)/(delta_time004)/
(ifspeed013+1)*100.0;
if (ROTATE.utilization>=40);
  utilization_score=4.0;
  utilization40to100=1.0;
else if ((ROTATE.utilization>=20) && (ROTATE.utilization<40));
  utilization_score=3.0;
  utilization20to40=1.0;
else if ((ROTATE.utilization>=10) && (ROTATE.utilization<20));
  utilization_score=2.0;
  utilization10to20=1.0;
else;
  utilization_score=1.0;
  utilization0to10=1.0;
endif;
```

In this example:

- A new column named utilization in the output table is computed from an expression that uses inoctet, outoctet, delta_time, and ifspeed values from the rate input table. (The +1 in the expression ifspeed013+1 is used to prevent a potential divide-by-zero condition. The ROTATE. notation in the next statement is used to refer to the new column in the output table.)
- A new column named utilization_score is computed and assigned a graded value as follows:
  - 4.0 for utilization percentages in the range 40 - 100%.
  - 3.0 for utilization percentages in the range 20 - 40%.
  - 2.0 for utilization percentages in the range 10 - 20%.
  - 1.0 for utilization percentages less than 10%.

TRENDsum then uses this graded value in the utilization_score column to compute a single, average GOS value (allgos) across all LAN interfaces combined, as shown in the following formula:

```
column:allgos = ((utilization_score*0.60) +
(discard_pct_score*0.25) +(error_pct_score*0.15)):avg
```

The allgos value appears in the LAN Connectivity Executive Summary report shown in Figure 9-2.

Other Data Aggregation Functions

Once TRENDit has cleaned the raw input data and TRENDstep has, optionally, conditionally processed the rows of the resulting rate table, TRENDsum can be used to summarize the data by any defined element property, by time, or by element property and time; and compute multiple, complex statistics about the summarized data. Specifically, TRENDsum can compute the following statistics:

- Total, average, weighted average, standard deviation, median, minimum, and maximum.
- ta_period and delta_time when minimum and maximum values occur.
- 90th, 95th, and 98th percentiles.
- 30-, 60-, and 90-day resource utilization forecasts.
- Days-until-threshold forecasts.

While TRENDit generates a separate row for each statistic, TRENDsum generates only one row for all the statistics you select to compute. Each statistic appears in a separate column in the row. TRENDsum generates separate rows for each of the following grouping categories:

- Any property that is defined for an element. Examples of element properties include the device itself (a router, for example), as well as properties of that device (for example, Ethernet, FDDI, token ring, serial 56, serial T1 interfaces; the speed of an interface; the location of an interface, such as Sales Department; or the client to whom the resource is dedicated, such as the XYZ Company).

- Time. The time categories by which you can group collected data include:
  - Hour, day, week, month, quarter, and year.
  - Day of the week (for example, by all Mondays, Tuesdays, and so on). This grouping results in 7 rows in the output table for each element property being summarized.
  - Day of the week by hour (for example, 1:00 a.m. on Mondays, 2:00 a.m. on Mondays, up to 24:00 p.m. on Mondays; 1:00 a.m. on Tuesdays, 2:00 a.m. on Tuesdays; and so on). This grouping results in 7 * 24 rows in the output table for each element property being summarized.
  - Hour of day (for example, 1:00 a.m., 2:00 a.m., 3:00 a.m., and so on). This grouping results in 24 rows in the output table (one for each hour of the day) for each element property being summarized.

The statistics that TRENDsum can compute and the grouping options available are much richer than the ones available with TRENDit. Therefore, the TREND ReportPacks in TREND 3.5 and later releases use TRENDsum rather than TRENDit to create the needed summary tables.

TRENDsum requires cleaned data as input. “Cleaned” in this context means that raw SNMP counter data has been turned into delta data. Thus, the input source can be a rate table produced by TRENDit or some other facility, a TRENDstep file, or another TRENDsum file.

In addition to the requested statistics, other columns in the TRENDsum output table include:

Column Name	**Description**
ta_period	Identifies the beginning of the grouping period. For example, if you are grouping by hour, all samples collected between 8:00 a.m. and 9:00 a.m. are assigned the ta_period value of 8:00 a.m.
ta_samples	Contains the count of input samples used to compute the row.

Forecasting

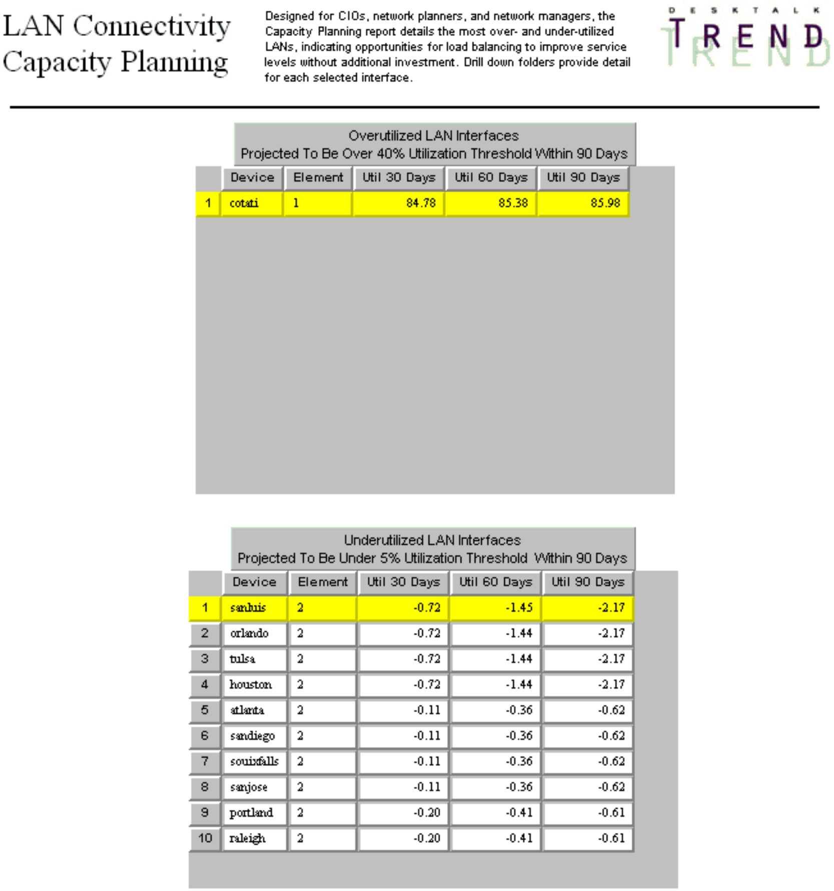
TRENDsum also enables you to forecast resource usage for 30-, 60-, and 90-day periods and compute the number of days until a specified usage threshold is reached (days-to-threshold forecasting).

The 30-, 60-, and 90-day forecasts answer the question: Where will the property being forecast (e.g., %Utilization) be in 30, 60, or 90 days?

The days-to-threshold (DTT) forecast answers the question: In how many days will the property being forecast be at the threshold value?

Forecasts are used in the Capacity Planning and Forecast reports supplied with the TREND ReportPacks. Figure 9-3 shows a sample LAN Connectivity Capacity Planning report.

In this report, TRENDsum formulas for computing 30-, 60-, and 90-day forecasts are used.



LAN Connectivity
Capacity Planning

Designed for CIOs, network planners, and network managers, the Capacity Planning report details the most over- and under-utilized LANs, indicating opportunities for load balancing to improve service levels without additional investment. Drill down folders provide detail for each selected interface.

Overutilized LAN Interfaces
Projected To Be Over 40% Utilization Threshold Within 90 Days

	Device	Element	Util 30 Days	Util 60 Days	Util 90 Days
1	cotati	1	84.78	85.38	85.98

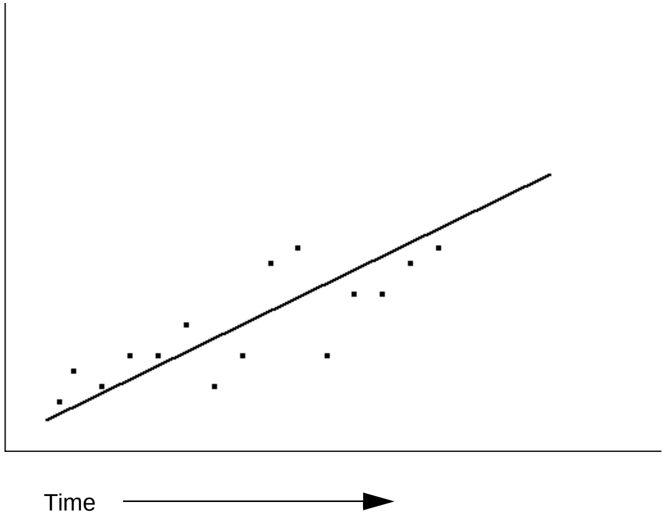
Underutilized LAN Interfaces
Projected To Be Under 5% Utilization Threshold Within 90 Days

	Device	Element	Util 30 Days	Util 60 Days	Util 90 Days
1	sanluis	2	-0.72	-1.45	-2.17
2	orlando	2	-0.72	-1.44	-2.17
3	tulsa	2	-0.72	-1.44	-2.17
4	houston	2	-0.72	-1.44	-2.17
5	atlanta	2	-0.11	-0.36	-0.62
6	sandiego	2	-0.11	-0.36	-0.62
7	souixfalls	2	-0.11	-0.36	-0.62
8	sanjose	2	-0.11	-0.36	-0.62
9	portland	2	-0.20	-0.41	-0.61
10	raleigh	2	-0.20	-0.41	-0.61

Figure 9-3: LAN Connectivity Capacity Planning Report

In the Capacity Planning and Forecast reports, all TREND forecasting is based on 95th percentile values computed on daily summary input tables that TRENDsum has created. In these reports, the 95th percentile values for a particular element property (or expression, such as %Utilization) over time are used for forecasting, because percentile values eliminate spikes from the collection samples.

TRENDsum forecasting calculations are based on a least squares linear regression, which means that the calculations attempt to draw a straight line through a set of data points collected over time, as shown in the following diagram:



Each data point represents a daily 95th percentile value for the property being forecast. You want to look at peak utilization for planning purposes, so TRENDsum uses a daily 95th percentile value, which is a near-maximum value.

For a DTT calculation, you have these cases:

- You will hit the threshold in the calculated number of days.
- You are already beyond the threshold.
- You are never going to hit the threshold (the line slopes downward, away from the threshold value).
- You are a long distance from the threshold. If the threshold is too far away, the forecast is meaningless.

If the slope is negative (decreasing, sloping downward), the result returned for DTT is a null value. Likewise, if the threshold is more than 1,000 days away, the result returned is null.

The formulas that TRENDsum uses to compute all forecasts are given in the section on the TRENDsum utility in “Man Pages” on page A-1.

Baselines

A *baseline* provides a basis for comparing recent performance with past performance, for example, for comparing last week’s performance with performance for the last two months.

You use TRENDsum to create a baseline. TRENDsum enables you to compute a baseline for any defined element property. You can also baseline a computed metric, such as utilization, which is computed as an expression that refers to one or more element properties.

TRENDsum enables you to choose how long you want the baseline period to be (the TRENDsum -y option). The default is 42 days (six weeks).

TRENDsum creates a *rolling baseline* table. To perform rolling-baseline aggregation, use the -y option on the command line to specify the length of the baseline period in days.

In a rolling baseline, the baseline table contains *n* rows; each row represents the same statistic (or statistics) computed for the immediately preceding number of days in the baseline period. Here is how a rolling daily baseline is computed if you use a 42-day baseline period:

1. At the end of day 1, the baseline statistic is computed from all the sample data collected for day 1 and stored in a summary row for day 1. For example, assume you are computing an average value to be used as a baseline. If 96 sam-

ples are collected for day 1 (one sample every 15 minutes), the values for all samples are totaled and divided by 96 to compute the average for the day 1:

Day	**Baseline Statistic**
1	a_1

2. At the end of day 2, the baseline statistic is recomputed from all the sample data collected for days 1 and 2 and stored in the summary row for day 1, thus *replacing the existing day 1 value*. For example, if 96 samples are collected for day 2, those 96 values and the 96 values collected on the previous day (day 1) are totaled and divided by the sample count (now 192—96 for day 1 plus 96 for day 2) to arrive at value a_2:

Day	**Baseline Statistic**
1	a_2

3. In the same fashion, the baseline statistic is recomputed at the end of day 3 from all the sample data collected for days 1, 2, and 3 and stored in the summary row for day 1, again replacing the existing day 1 value. At this point, the value (a_3) for the baseline summary row represents a 3-day aggregate:

Day	**Baseline Statistic**
1	a_3

4. The same processing is repeated at the end of each day until a baseline value has been recalculated for each day of the rolling baseline period (42 days, in this example). Thus, at the end of 42 days, a single baseline value has been computed based on data collected on the previous 42 days:

Day	**Baseline Statistic**
1	a_{42}

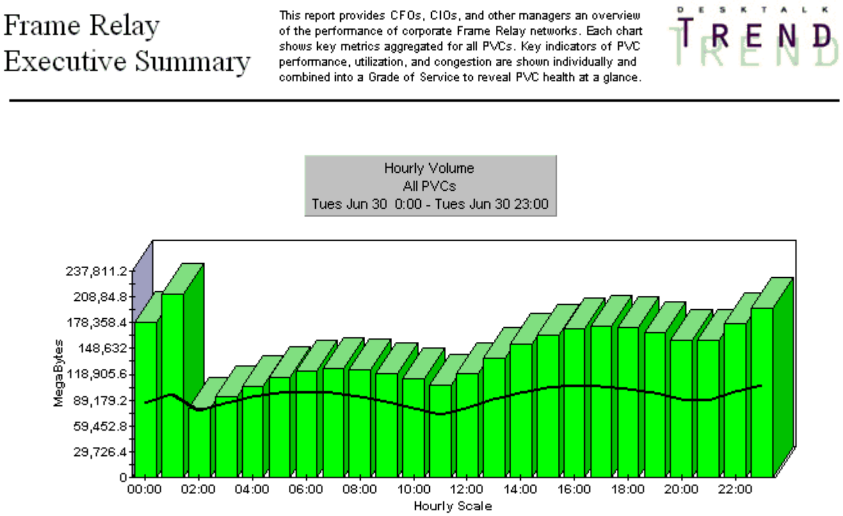
This day 1 baseline value, because 42 days of data is used to compute it, is significantly more refined than a value where only one day's samplings are summarized to compute the baseline statistic for day 1.

5. When the number of days in the rolling baseline period is reached, processing continues in the fashion described above. However, the rolling baseline value that is computed from the preceding 42 days of samples is appended to the table. The existing baseline is no longer replaced. Thus, rows are added to the rolling baseline summary table as follows:

Day	**Baseline Statistic**	**Computed from**
1	a_1	Days 1 - 42
2	a_2	Days 2 - 43
3	a_3	Days 3 - 44
4	a_4	Days 4 - 45
n	a_n	Days *n* - (*n*-42)

The Executive Summary reports in the TREND ReportPacks compare a resource usage for each hour of the day to an hourly baseline that has been computed for the resource. The concept for computing an hourly baseline is the same as the one for computing a daily baseline. However, the hourly baseline table has one row for each hour of each day computed for a sliding 42-day baseline period as described above.

Figure 9-4 shows an excerpt from the Frame Relay Executive Summary report.



Frame Relay Executive Summary

This report provides CFOs, CIOs, and other managers an overview of the performance of corporate Frame Relay networks. Each chart shows key metrics aggregated for all PVCs. Key indicators of PVC performance, utilization, and congestion are shown individually and combined into a Grade of Service to reveal PVC health at a glance.

Figure 9-4: Sample Frame Relay Executive Summary Report (Baselines)

In this report, the hourly baseline is drawn as a line over the hourly PVC volume. The baseline is computed for each hour of each day for the preceding 42-day window.

Ranking

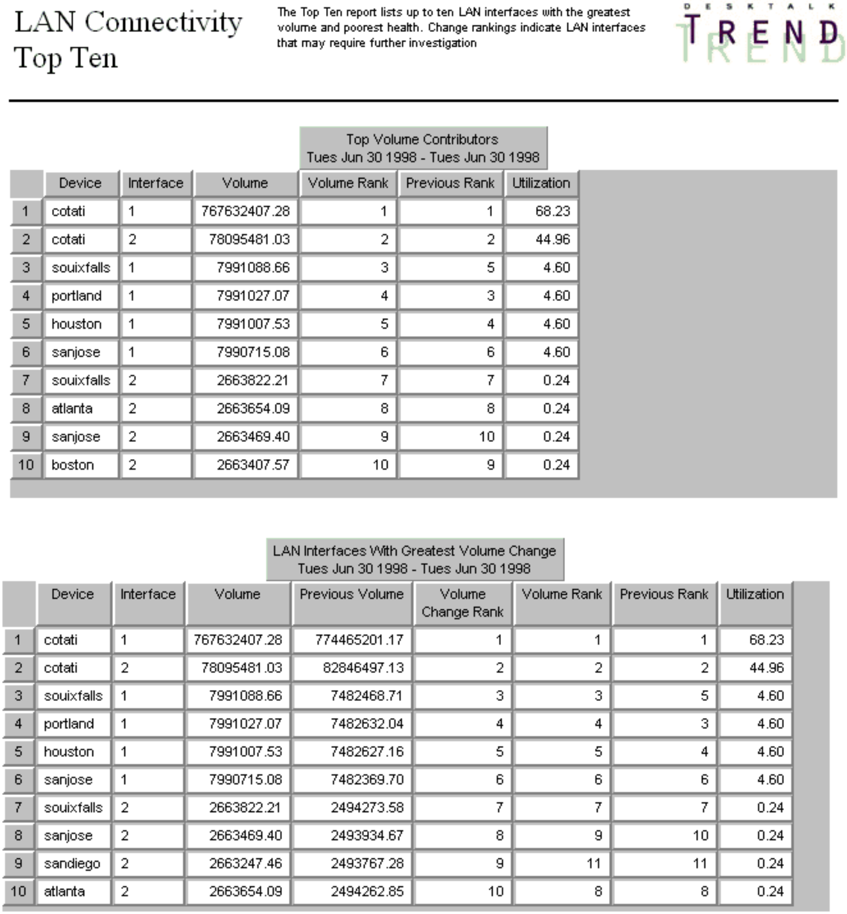
TRENDrank applies ranking criteria to summarized data and stores the results in the TREND database. Ranking enables you to chart the volatility of a given metric for a given element. As an example, use utilization and congestion among 100 PVCs to answer the following types of questions:

- Which ten PVCs rank the highest in terms of utilization and congestion on any given day?
- How often does the same element appear in the top ten in terms of utilization and congestion?
- Are there significant variations in element rank from day to day (delta rank)? A large delta rank indicates a volatile element, which you should probably investigate. A relatively small delta rank indicates stable element performance.

TRENDrank ranks the columns of a daily summary table that TRENDsum has created and stores the results in a ranking table in the TREND database. The ranking table has one row per day for each element that is being ranked, and the following seven columns:

- Timestamp (for day1, day2, day3, and so on).
- ID of the element being ranked.
- Rank (1, 2, 3 and so on).
- Daily summary value of the metric being ranked.
- Rank for the element on the previous day.
- Delta rank. This item represents the difference in the element's rank between yesterday and today. That is, if the element was ranked 8 yesterday and 5 today, delta rank is 3.
- Previous delta rank.

Data that is ranked by TRENDrank appears in the Top Ten reports in the TREND ReportPacks. Figure 9-5 shows an excerpt from the LAN Connectivity Top Ten report:



LAN Connectivity Top Ten

The Top Ten report lists up to ten LAN interfaces with the greatest volume and poorest health. Change rankings indicate LAN interfaces that may require further investigation

Top Volume Contributors
Tues Jun 30 1998 - Tues Jun 30 1998

	Device	Interface	Volume	Volume Rank	Previous Rank	Utilization
1	cotati	1	767632407.28	1	1	68.23
2	cotati	2	78095481.03	2	2	44.96
3	souixfalls	1	7991088.66	3	5	4.60
4	portland	1	7991027.07	4	3	4.60
5	houston	1	7991007.53	5	4	4.60
6	sanjose	1	7990715.08	6	6	4.60
7	souixfalls	2	2663822.21	7	7	0.24
8	atlanta	2	2663654.09	8	8	0.24
9	sanjose	2	2663469.40	9	10	0.24
10	boston	2	2663407.57	10	9	0.24

LAN Interfaces With Greatest Volume Change
Tues Jun 30 1998 - Tues Jun 30 1998

	Device	Interface	Volume	Previous Volume	Volume Change Rank	Volume Rank	Previous Rank	Utilization
1	cotati	1	767632407.28	774465201.17	1	1	1	68.23
2	cotati	2	78095481.03	82846497.13	2	2	2	44.96
3	souixfalls	1	7991088.66	7482468.71	3	3	5	4.60
4	portland	1	7991027.07	7482632.04	4	4	3	4.60
5	houston	1	7991007.53	7482627.16	5	5	4	4.60
6	sanjose	1	7990715.08	7482369.70	6	6	6	4.60
7	souixfalls	2	2663822.21	2494273.58	7	7	7	0.24
8	sanjose	2	2663469.40	2493934.67	8	9	10	0.24
9	sandiego	2	2663247.46	2493767.28	9	11	11	0.24
10	atlanta	2	2663654.09	2494262.85	10	8	8	0.24

Figure 9-5: Sample LAN Connectivity Top Ten Report (Ranking)

Summary

As introduced in this chapter, the TRENDit, TRENDstep, TRENDsum, and TRENDrank utilities produce and process cleaned data. They provide the means for removing invalid or incomplete samples, leveling performance spikes, creating meaningful aggregation levels, and computing statistics that provide useful measurements of enterprise health.

"Man Pages" on page A-1 describes the syntax and usage of these utilities in detail.

10 TRENDbuild

TRENDbuild is a report building utility that defines graphical and tabular TREND reports. It allows you to specify data to be included in a report, assign aliases to fields being reported on, create expressions that derive calculated values from stored data for inclusion in a report, and save the report definition as a query file. TRENDbuild is used to create new reports for TRENDgraph, TRENDsheet, and Grade of Service applications.

Main Display and Menu System

The main **TRENDbuild** window (Figure 10-1) can be invoked from the Tools menu of the main TREND window or from the TRENDgraph or TRENDsheet applications.

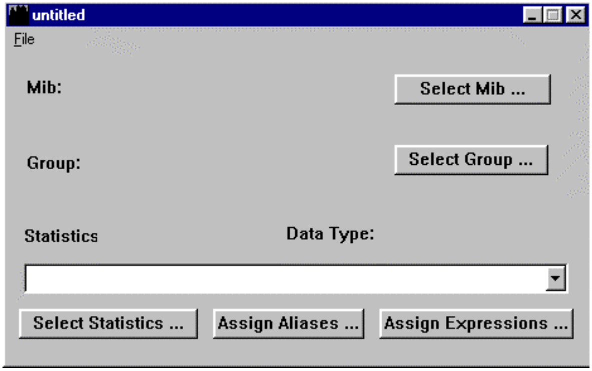
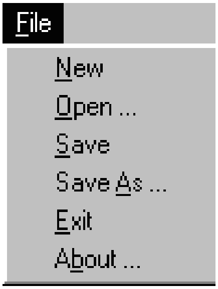


Figure 10-1: TRENDbuild Main Window

The File menu (shown below) is used to create, access, and exit reports.



About Data and Data Tables in TREND

TRENDbuild creates and/or edits report templates which are comprised of files containing report set-up definitions and a specially formatted SQL query. TRENDbuild requires neither SQL nor network expertise to use it. In order to make full use of TRENDbuild's functionality, however, some basic understanding of the data and data formats available in the TREND database is required.

Each group of MIB objects, or miblet, being collected has a corresponding raw data table in the TREND database. Raw data tables in TREND are populated with timestamp, constants, statistics and other values. Statistics may be counters or gauges. A counter is a raw count of a statistic since a device was rebooted. A gauge is an instantaneous measurement. Examples include percent CPU utilization, percent buffer utilization, or the number of buffers used at a point in time. Gauge values apply to a particular instant in time, while counters are cumulative over time.

TREND includes a module called TRENDit that aggregates or rolls up raw data. TRENDit runs on a batch-mode basis. By default TRENDit will run once each night rolling up all data tables. For each raw data table, TRENDit will create (or add rows to) a rate table, an hourly table, a daily table, a weekly table, and a monthly table based upon the amount of valid data available for aggregation. TRENDbuild equates a data table to a group.

Note: Raw data tables contain values as collected from the polled devices. Most performance-related values are counters. Consequently, most performance reports are based on roll-up tables, not raw tables.

All TREND data tables have a similar construction, consisting of header, data and footer fields. All headers contain identifiers for the source of the data, a received timestamp, a requested timestamp, and a delta time value.

Data fields correspond to objects in the associated miblet. The footer fields include indexing information and for hourly, daily, weekly, and monthly tables, a timestamp marking the beginning of the aggregation period (ta_period) and the number of raw data samples collected during the aggregation period (ta_samples).

Note: All data tables contain a delta time value. In raw data tables, delta time is the length of time since the data source was rebooted, in units of one hundredths of a second (i.e., sysuptime). In rollup tables the delta time value is the time between samples in units of seconds. For reporting purposes, a column called ta_sysuptime is added in all roll-up tables. It

reflects the time since the data source was rebooted in hundredths of a second, at the end of the measured time.

Each raw data table and its associated roll-up tables have a corresponding key table. Key tables are used to maintain a unique index-like value for each unique data source (e.g., a router name and interface number (ifIndex) for MIB-II-ifEntry data). Key tables also contain a description column where information about the data source can be placed. In addition, some key tables contain additional configuration and descriptive information about the data source (e.g., RMON key tables contain interface type and speed information). These values can be selected by TRENDbuild for use in reporting.

Rate table rollup primarily deals with counter variables. The structure for the raw and rate tables is virtually identical. Rate tables are generated by selecting pairs of rows from the raw data tables (successive data points). The delta between the points is calculated and the result inserted into the rate table. The values for most columns in the new rate table row are the same as the more recent of the two samples.

Note: The term rate table is something of a misnomer as used here. Rate tables can be used to calculate rates by dividing counter variable deltas by delta time. Unless captured as gauges, no rates are directly stored in the rate table.

In addition to aggregating raw data into rate tables, TREND rolls up rate data into hourly, daily, and weekly tables.

The process that occurs between each level of rollup is virtually identical. In each case (hourly, daily, weekly, and monthly) four rows are created in the corresponding table. These four rows correspond to the aggregation that has been performed on the data: average (avg), minimum (min), maximum (max), or total (tot). Hourly tables are calculated from rate tables. Daily tables and weekly tables are calculated from hourly and daily tables respectively using the *total* row from that table. Monthly tables are calculated from daily tables.

Statistic	Counters	Gauges	Other Types
Average (avg)	Sum of values divided by total delta time (i.e., units per second)	Average value of samples in interval	Value from the latest timestamp row in interval
Minimum (min)	Value from sample with minimum rate (in units per second)	Minimum value of samples in interval	Value from the latest timestamp row in interval
Maximum (max)	Value from sample with maximum rate (in units per second)	Maximum value of samples in interval	Value from the latest timestamp row in interval
Total (tot)	Total of values for all samples in interval	Average value of samples in interval	Value from the latest timestamp row in interval

Creating a New Report

Defining a new report focuses on the data to be reported on and what it should be called in the report. The series of steps to follow are:

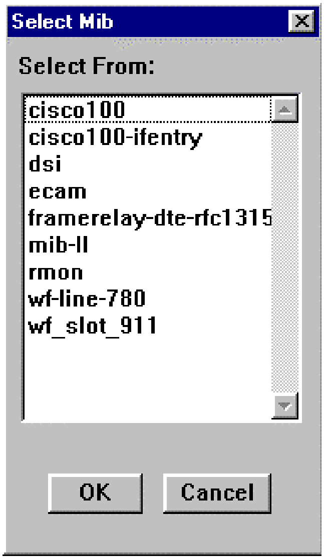
1. Select a MIB
2. Select a group (usually a set of MIB objects, i.e. miblet)
3. Select statistics
4. Assign aliases
5. Create expressions

6. Save the report

If there is already a report open in TRENDbuild, you can clear it from the current context by selecting the **New** option under the File menu.

Selecting MIBs and Groups

A category is an organizational level for collections or groups of data. In general, a category is equivalent to an SNMP MIB file. The next level of data organization under a category is a groups. A group includes a collection of related raw data tables and the rollup tables derived from the raw tables. To choose a category, click on the Select Mib button to bring up the **Select Mib** window:



Select Mib

Select From:

cisco100
cisco100-ifentry
dsi
ecam
framerelay-dte-rfc1315
mib-II
rmon
wf-line-780
wf_slot_911

Figure 10-2: Select MIB Window

Once a category is chosen, its name appears in the Mib box. Scroll through the entries in the window to find the one you want.

- Select a category by clicking on a category name listed, then clicking on the OK button, or simply by double-clicking on the category name.
- Click on the Cancel button to quit out of the window without making a selection.

The name of the category you choose will appear in the Category box in the main TRENDbuild window.

The second step in defining a report is to select the group which contains the statistics to be reported on.

To select a group, click on the **Select Group** button in the TRENDbuild Main window (Figure 10-1). This will bring up the Select Group window:

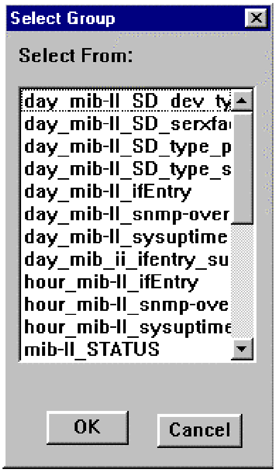


Figure 10-3: Select Group Window

A group is the specific type of data to be used in the report. It is a collection of statistics which are stored in a single table in the database. Just as a category can be thought of as an SNMP MIB, a group can be thought of as a set of MIB objects. Initially, the database contains a single table or group corresponding to the raw MIB object data. As data is aggregated, a set of roll-up tables or groups is created.

Note: EXAMPLE: If the only group of statistics being collected based on the MIB-II file was the ifEntry table, once rollup had occurred you would see the selection in the Select Group window as shown in Figure 10-3. **See** "About Data and Data Tables in TREND" on page 10-2 **for a description of the contents of these roll-up tables and groups.**

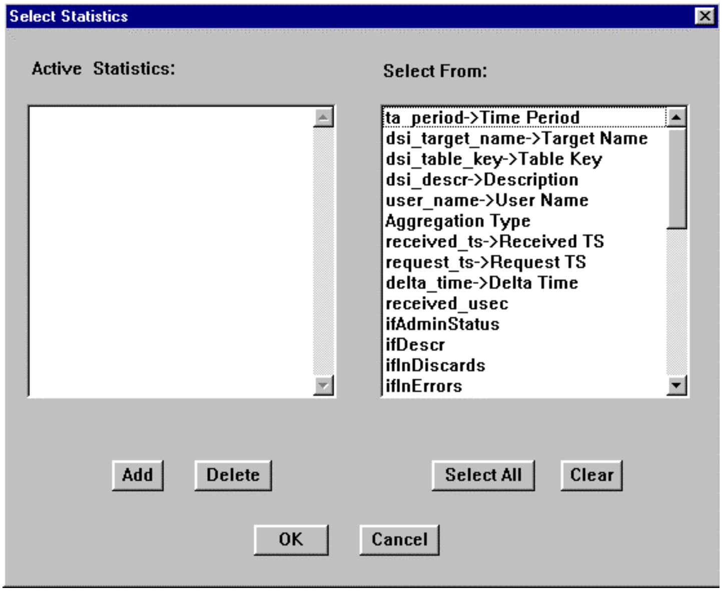
Scroll through the entries in the Select Group window to find the one you want.

- Select a group by clicking on a group name listed, then clicking on the **OK** button (or double-click the group name).
- Click on the **Cancel** button to quit out of the window without making a selection.

The name of the group you choose will appear in the **Group** box in the main TRENDbuild window.

Selecting Statistics

Statistics are individual data fields in a group. Once the category and group are chosen, you can build the list of which statistics in the group will be in the report using the Select Statistics window. Bring the window up by clicking on the **Select Statistics** button in the main TRENDbuild window:



Select Statistics

Active Statistics:

Select From:

ta_period->Time Period
dsi_target_name->Target Name
dsi_table_key->Table Key
dsi_descr->Description
user_name->User Name
Aggregation Type
received_ts->Received TS
request_ts->Request TS
delta_time->Delta Time
received_usec
ifAdminStatus
ifDescr
ifInDiscards
ifInErrors

Figure 10-4: Select Statistics Window

Note: Before selecting specific statistics it is important to consider the unit of measurement for each statistic. Instead of selecting and using statistics in their natural form, you may instead want to create expressions. "Assigning Aliases and Creating Expressions" on page 10-10 **defines the units of measurement for statistics in raw and roll-up tables.**

Select specific statistics to include by clicking on their names in the **Select From** box. Selected names will be highlighted. You can deselect a name by clicking on it a second time. You can select all the statistics at once by clicking on the **Select All** button. You can deselect all the selected statistics at once by clicking on the **Clear** button.

Once you have selected the statistics you want, include them by clicking on the **Add** button. Their names will appear in the Active Statistics window.

Note: Their names will continue to appear in the Select From box even after they are selected. If you want to exclude one or more active statistics, click on their names in the Active Statistics window to highlight them, then click on the Delete button. Their names will be removed from the window.

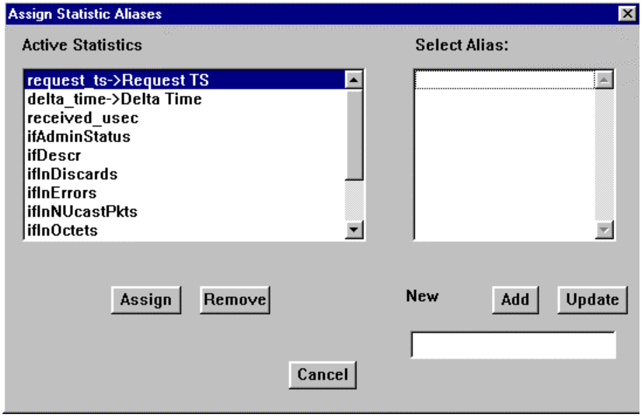
Add and delete statistics until the list in the Active Statistics box is the way you want it;

- Apply their names to the report by clicking on the **OK** button.
- Quit out of the window without saving your changes by clicking on the **Cancel** button (if you **OK** prior to clicking **Cancel** changes are effective).

The names of the selected statistics will appear in the **Statistics** box in the main TRENDbuild window. You can scroll through the names in the box by clicking on the down arrow in the box, then using the scroll bar in the box to move through the list. If you click on the name of a statistic in the box, its data type will be displayed in the **Data Type** box.

Assigning Aliases and Creating Expressions

The names of statistics shown in the Select Statistics window are used to label their associated data in reports. You can override these names with more readable and descriptive labels by assigning aliases to the statistics. This is done in the Assign Statistic Aliases window. To bring up the window, click on the **Assign Aliases** button in the main TRENDbuild window:



Assign Statistic Aliases

Active Statistics

request_ts->Request TS
delta_time->Delta Time
received_usec
ifAdminStatus
ifDescr
ifInDiscards
ifInErrors
ifInNUcastPkts
ifInOctets

Select Alias:

Figure 10-5: Assign Statistic Aliases Window

The Active Statistics box in the window displays the currently active statistics as set-up in the Select Statistics window. You can assign an alias to any of these names.

Create an alias by typing it in the **New Alias** box. The alias will appear in reports exactly the way you type it here. When it is the way you want it, make it available for assignment to a statistic by clicking the Add button. The alias string will appear in the **Select Alias** box.

You can modify an alias string by clicking on it in the **Select Alias** box, which will cause the string to appear in the **New Alias** box. Make your changes, then click on the **Update** button. The new value will replace the original in the **Select Alias** box.

Assign an alias string to a statistic by clicking on a statistic in the **Active Statistics** box to highlight it, then click on an alias string in the **Select Alias** box to highlight it. Now click on the **Assign** button. The names are now associated. A statistic with an alias assigned to it is displayed in the **Active Statistics** box in the format:

```
statistic-name -> alias-name
```

You can disassociate an alias from a statistic by clicking on the pair in the **Active Statistics** box then clicking on the **Remove** button. The ->*alias* part of the value will disappear from the statistic in the box.

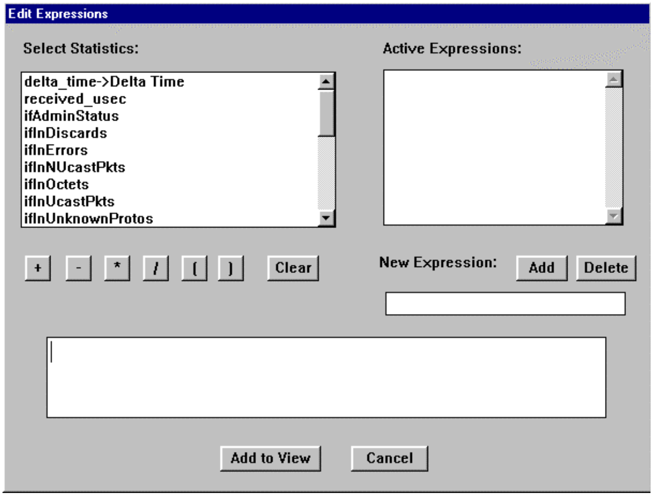
To apply an updated alias string to a statistic that already has an alias string, remove the alias from the statistic, then assign the new value. This will not be done automatically.

When you finish assigning aliases, quit the window by clicking on the **Cancel** button. The assigned aliases will appear with their associated statistics in the **Statistics** box in the main TRENDbuild window.

Note: The expression length allowed by TRENDbuild is 1024K.

An expression is a calculation that yields a derived value from existing data. Expressions created in TRENDbuild are used in the select clause of an SQL statement. The results are plotted or displayed like the values of a statistic.

Expressions are built in the Edit Expressions window:



delta_time->Delta Time
received_usec
ifAdminStatus
ifInDiscards
ifInErrors
ifInNUcastPkts
ifInOctets
ifInUcastPkts
ifInUnknownProtos

Figure 10-6: Edit Expressions Window

The Edit Expressions window lists all the statistics in the current group in the **Select Statistics** box in alphabetical order. All the statistics are available instead of just those activated in the Select Statistics window because any statistic may be needed for an expression, and because an expression is treated like a selected statistic by TRENDbuild.

Note: Although the Select Statistics box shows you the names of any assigned aliases, the statistics' actual names are used in expressions.

Build an Expression

Note: It is important to follow these steps in order.

1. Type in the expression's name in the **New Expression** box. The name you give the expression will be used as its name in the report. You cannot assign an alias to an expression, however, when you name the expression you are effectively giving it an alias.
2. When the name is the way you want it, click on the **Add** button. The name will appear in the **Active Expressions** box. Note that you cannot change an expression's name in the window, so make sure you give it the name you want.
3. To define the expression you just named, click on its name in the **Active Expressions** box. Click in the wide box near the bottom of the screen to enter the expression.
4. Build the expression by clicking on the statistic names and arithmetic operator buttons and entering numbers as necessary. You are responsible for correct syntax in the expression. TRENDbuild will not check your work.

Note: Values added to an expression, by clicking on statistic names or operator buttons, will be added to the end of the expression, regardless of the cur-

sor position. The use of parentheses is recommended for all expressions. Ensure that right and left parentheses match, for example: (((ifInOctets*8)/fSpeed)*100.0).

5. When the expression is defined the way you want it, click on the **Add to View** button to add it to the report definition. Do this before beginning to define another expression.

You can modify an existing expression. Click on its name in the Active Expressions window. You will see its name appear below the arithmetic operator buttons, and the expression text will appear in the box at the bottom of the window. You can delete the entire expression by clicking on the **Clear** button. Make any changes to the expression text, then click on the Add to View window to apply the changes.

When you finish defining expressions, quit the window by clicking on the **Cancel** button. The names of the expressions will appear in the **Statistics** box in the main TRENDbuild window in the form:

```
expression-name -> expression
```

If you select the expression, the word EXPRESSION will appear in the **Data Type** box in the window.

Modifying an Existing Report

An existing report can be modified by loading it into TRENDbuild, changing its definition and saving the changed version to disk. This is useful if you want to add or delete statistics in a report, change an alias or modify an expression. It is also useful as a way to create a series of reports based on common information.

To load an existing report into TRENDbuild, select the **Open** option under the **File** menu. This will bring up the File Selection window, which allows you to specify which report to open.

When the report is open, the various TRENDbuild windows will display its definition information. Modify these as necessary according to the procedures described in “Creating a New Report” on page 10-5.

To save the new version of the report, select the **Save** option under the File menu. You can create a new report using the modified definition, by selecting the **Save As** option under the **File** menu and saving the report with a new name in the File Selection window.

TREND

11 TRENDsheet

Tabular data displays are useful for reviewing lists of values in displayed in different orders or grouped for comparison. Tabular formats are especially useful for exception reporting, showing, for example, exceeded thresholds.

TRENDsheet provides these functions, and also allows selection of the node to report on, and which database to query for the data. TRENDsheet reports can be printed and data can be exported to an ASCII file for transfer to spreadsheets or other applications.

Main Display and Menu System

You invoke TRENDsheet from the Tools menu of the TREND main window. The TRENDsheet window appears initially with a blank display area and a menu bar providing access to TRENDsheet functions (see Figure 11-1):

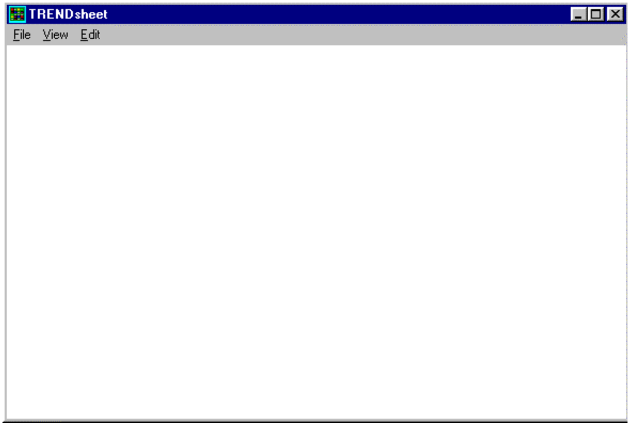
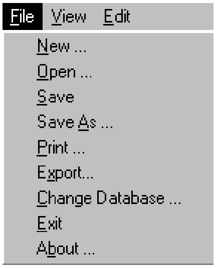


Figure 11-1: TRENDsheet Main Window

Menu Command Summary

File Menu



Use the **New** command to create a new tabular report (see "Creating a New Tabular Report" on page 11-5).

The **Open** command brings up pre-defined tabular reports (see "Opening an Existing Tabular Report" on page 11-4).

The **Save** option is used to store a previously saved report.

Save As is used to store new reports or to copy or move existing reports.

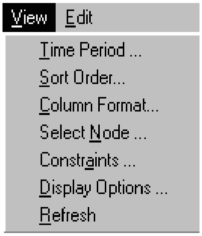
Tabular reports can be printed to a postscript printer using the **Print** command (see "Print" on page 1-17).

Export TRENDsheet data to word processors and other commercial applications (see "Exporting Data" on page 11-18).

The **Change Database** command provides access to a different SQL Server to display its version of the information presented in a report (see "Displaying Data from a Different Database" on page 11-11).

Quit the TRENDsheet application using the **Exit** command (see "Exit" on page 1-19 and "Selecting Statistics" on page 11-6).

The **About** command invokes TREND version information (see "About TREND" on page 1-19).



Time Period allows setting of the time range of data to be reported (see "Setting the Time Period" on page 11-12).

The **Sort Order** command allows sorting on different fields, or in ascending rather than descending order (see "Sorting the Data in the Report" on page 11-14).

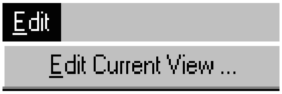
The **Column Format** option allows you to change the precision of numeric columns. (see “Formatting Columns in the Report” on page 11-15).

To focus a report on a specific node and one or all of its keys for the table, click **Select Node** (see “Select the Node and Key Being Reported” on page 11-9).

The **Constraints** command is used to specify thresholds to be used in exception reporting. Only values that meet certain criteria (e.g., exceed user-specified error rates) will be reported in the spreadsheet (see “Establishing Constraints” on page 11-12).

Use **Display Options** to turn the grid on or off, select font point size, set the maximum number of rows displayed, and specify titles and subtitles.

Selecting the Refresh option clears the report window and repopulates with the most current data available.



To modify the definitions setup through TRENDbuild, click the **Edit Current View** option (see “Modifying the Tabular Report” on page 11-9).

Opening an Existing Tabular Report

To open a pre-defined TREND report, select the Open command from the File menu. The query file will be opened via the File Selection window described in “The Open Command” on page 1-15.

Using an Existing Report as a Template

There may be times when you want to use one report as a template for another. To do this, open the existing report and click the Save As option under the File menu. This will bring up the File Selection window (see "The Open Command" on page 1-15), allowing you to write a new report file to disk. The new file can be modified to create a different report, then saved to preserve the changes.

Creating a New Tabular Report

New reports are defined using TRENDbuild (see "TRENDbuild" on page 10-1). You can invoke TRENDbuild from inside TRENDsheet by selecting the New option under the File menu:

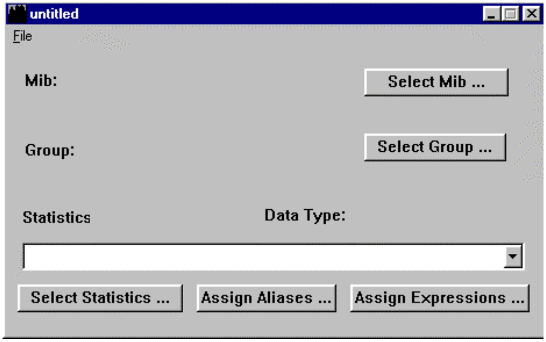


Figure 11-2: TRENDbuild Main Window

To finish defining the new report, you must set its display parameters. This includes which node and key or keys to display, column precision, specifying report titles, and setting the column order and column widths. Finally, Save the fully defined report.

Note: After modifying a newly created TRENDsheet to a file and saving it to a file, if you add a where clause to the entry, there must be a space before the word where to properly run the report.

Selecting Statistics

The first step in creating a new tabular report is to select a category (MIB) and group of data. Use the **Select Mib** and **Select Group** buttons in the TRENDbuild window. The Select Statistic box contains a list of statistics chosen, and expressions created, in TRENDbuild when the report was created. Select an expression using the Select Statistic box's down arrow button, then clicking on the desired statistic. Use the scroll bar to see all the available names.

EXAMPLE: *Select the Weekly RMON Host Group*

To create a report based on weekly rmon host data, first click Select Mib and select rmon from the Select Category pop-up box. Click OK.

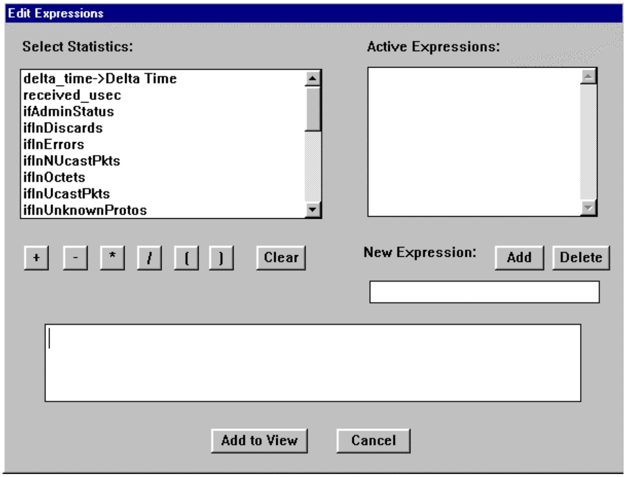
Next click Select Group and week_rmon_host-data from the Select Group pop-up box. Click OK.

The next step is building expressions to be displayed using the available statistics. Click **Assign Expressions** box in the TRENDbuild window to invoke the **Edit Expressions** window.

The **Edit Expressions** window provides:

- Selection of statistics that will be included in expressions to be displayed (upper left),
- Mathematical operators to be used in expressions (beneath Select Statistics box),
- Creation of expressions from statistics and operators (bottom),
- Naming and adding/deleting expressions, and

- Listing of **Active Expressions** assigned (upper right).



delta_time->Delta Time
received_usec
ifAdminStatus
ifInDiscards
ifInErrors
ifInNUcastPkts
ifInOctets
ifInUcastPkts
ifInUnknownProtos

Figure 11-3: Edit Expressions Window

EXAMPLE: *Creating an RMON Host Table Error Report*

To define a tabular report displaying packets transmitted, transmission errors, and error/transmission rates, open the **Edit Expressions:** window and click in the **New Expressions:** box and enter a name for the first expression to be created (e.g., **Packets**). Click **Add** to place the expression in the **Active Expressions** box. Click the new expressions name in the **Active Expressions** box to continue (make sure it is highlighted).

Select **hostOutPkts** from the Statistics box. It will appear in the **Edit Expressions** window. Click **Add to View** to save the Packets statistic.

Repeat the above steps for the expression **Errors** using the **hostOutErrors** statistics.

In the same manner, define **Error Rate** expression and assign its expression:

```
((hostOut Errors)/(hostOutPkts)*100.0).
```

It is important that left and right parentheses are paired.

When all expressions have been created, click **Cancel** to exit the **Edit Expressions** window.

Once you have finished defining the basic report in TRENDbuild, you must save the query file you have created. Use either the **Save, Save As,** and **Exit** options on the TRENDbuild File Menu to invoke the **File** Selection window. You will be prompted to enter a file name under which the file will be stored. The file will then be automatically loaded into TRENDsheet. After saving the report, select **Exit** to quit TRENDbuild and return to TRENDsheet. The new tabular report will be displayed.

If there are more columns than can be displayed at one time in the TRENDsheet window, use the scroll bar at the bottom of the window. Fixed columns (see “Formatting Columns in the Report” on page 11-15) are always displayed in the window while the remaining columns will change as you scroll horizontally.

The default table length for a TRENDsheet report is 50 rows. The maximum length is 2,000 rows.

The report title is displayed centered at the top of the table grid area. It has three components:

file name the report file name, without the .qss extension.

node name the name of the node selected in the Select Node window.

key the value of the key chosen for the selected in the Select Node window (Figure 11-4). The key is shown in parenthesis ().

If you have selected **all nodes** from the **Node** box in the Select Node window (Figure 11-4), only the report name will appear.

Modifying the Tabular Report

To modify the definitions setup through TRENDbuild, click the Edit Current View option under the Edit button. This will put you into TRENDbuild, in the context of the current report. That is, the TRENDbuild category, group, statistics, aliases and expressions windows will contain the values used by the report being modified. When you finish making changes in TRENDbuild, you will need to reload the report into TRENDsheet using the Open option under the File button for those changes to be reflected in the report on your screen.

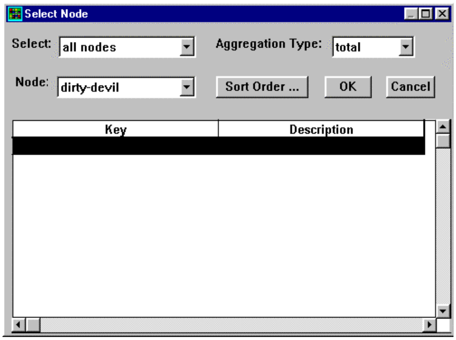
If you made any changes to the report definition before requesting Aliases/Expressions, TRENDsheet will ask if you want to save those changes before proceeding.

Specifying Data to be Included in the Report

Once a basic report has been defined and displayed, specific nodes and keys can be selected, the database can be changed, and a time period can be specified.

Select the Node and Key Being Reported

When you first run the new report, TRENDsheet will display the data for all nodes, and all keys for each node. You can focus the report on a specific node, and one or all of its keys for the table. Click the **Select Node** option under the **View** button to bring up the Select Node window:



Select: all nodes

Aggregation Type: total

Node: dirty-devil

Figure 11-4: Select Node Window

The **Node** box in the window lists all nodes that have data in the table being reported on in the database. It also gives you the option **all nodes,** which is the *default*. If you choose **all nodes** you automatically see all keys for all nodes in the report.

The **Aggregation Type** box is used to select the minimum, maximum, total, and average, of all types of aggregation for rolled-up tables (not raw tables).

Click the **Sort Order** button to bring up the Sort Order window, which allows you to select the order that the TRENDsheet data is sorted by, (see "Sorting the Data in the Report" on page 11-14), for more detailed information on how to sort.

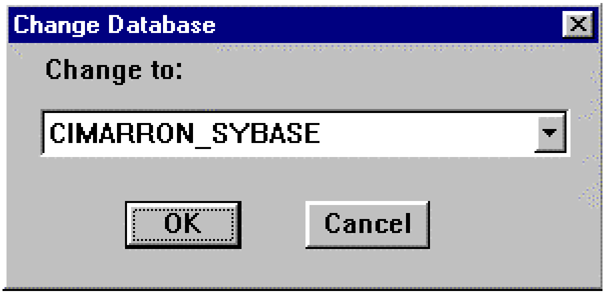
If you select a particular node in the Node box by clicking on its name, the Key box will display a list of all keys in the table for the selected node. Select the desired key from the Key box.

- When you have chosen the node/key pair you want, click the **OK** button to have the report only display values for that pair.
- Click **Cancel** to quit out of the window to without changing the current node/key pair values.

The node/key pair you choose will be saved as part of the report definition file so they will automatically be used the next time you run the report.

Displaying Data from a Different Database

Access a different SQL Server to display its version of the information presented in this report. Click the Change Database option under the File button to bring up the Change Database window:



Change to: CIMARRON_SYBASE

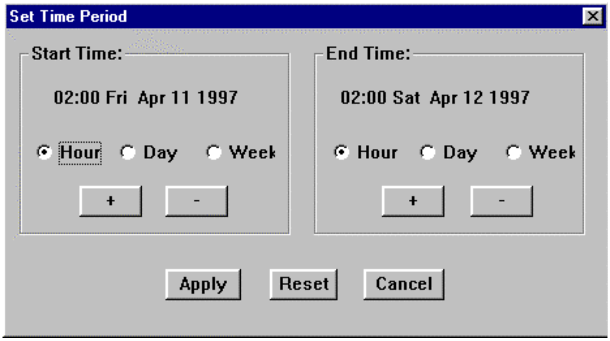
Figure 11-5: Change Database Window

The name of your current database is displayed in the Change to box. To change to a different server, click the down arrow in the Change to box to see a list of the Sybase SQL Servers known to this machine. The values presented here are read from the `$SYBASE/interfaces` file. Select the name of the server you want to access in the report.

- Click the **OK** button to change to the selected database.
- Click **Cancel** to quit out of the window without changing servers.

Setting the Time Period

TRENDsheet allows you to set a Start Time and an End Time for displaying data for reports. If you select Time Period from the View Menu of TRENDsheet, the Set Time Period window is displayed.



Set Time Period

Start Time:

02:00 Fri Apr 11 1997

● Hour ○ Day ○ Week

End Time:

02:00 Sat Apr 12 1997

● Hour ○ Day ○ Week

Figure 11-6: Set Time Period Window

Change the Start and End times by selecting the button that corresponds to the portion of the time you wish to set (i.e., **Hour, Day, or Week**). Next, click either the + or - buttons to increase or decrease this value.

- Click the **Apply** button to have the sort order you specified take effect.
- Click the **Reset** button to return to the default sort order.
- Click the **Cancel** button to quit out of the window without changing the current sort order.

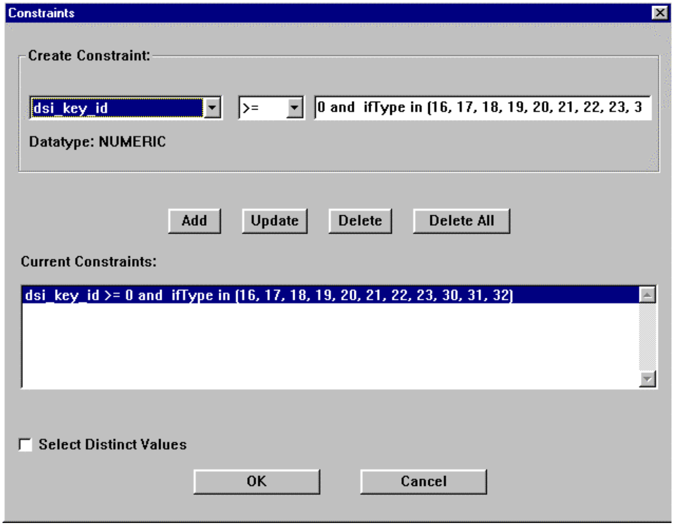
Establishing Constraints

TRENDsheet provides exception reporting capability. By selecting a statistic, a comparison operator, and a comparison value, tabular reports can be created that will show only the exception conditions specified.

First select a statistic using the pull down box at the left-hand side of the **Create Constraint:** box. The second pull down box contains the following operators: = (equal to), <= (less than or equal to), >= (greater than or equal to), < (less than), > (greater than), != (not equal to), !< (not less than), !> (not greater than), and **like** (compares the first value to a following character string).

Next, enter any value or character string in the box at the right. This entry will be compared against the value of the statistic specified using the operator selected.

Click the **Select Distinct Values** box to report on only the first instance where a constraint is met. Finally, click **Add** to place the expression in the **Current Constraints** box and click **OK** to exit.



dsi_key_id

>=

0 and ifType in (16, 17, 18, 19, 20, 21, 22, 23, 3

Datatype: NUMERIC

dsi_key_id >= 0 and ifType in (16, 17, 18, 19, 20, 21, 22, 23, 30, 31, 32)

□ Select Distinct Values

Figure 11-7: Constraints Window

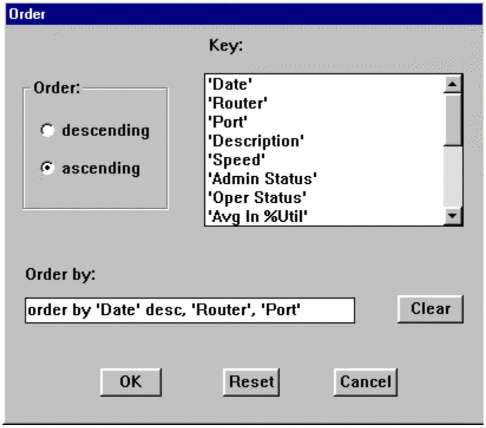
Formatting the Report

The look of tabular reports can be modified by sorting data and reformatting column parameters.

Sorting the Data in the Report

TREND data is time related, therefore TRENDsheet’s default sort order is by target name, table keys for a particular target name, and received timestamp (i.e. newest to oldest) for a particular table key, in that order. Since most reports will be run for a particular device, and a specific key for that device, the timestamp becomes the significant sort field.

There may be reports that you would rather sort on a different field, or in ascending rather than descending order. This is accomplished through the Order window, which is displayed by clicking on the Sort Order option under the View button:



Order:
○ descending
● ascending

'Date'
'Router'
'Port'
'Description'
'Speed'
'Admin Status'
'Oper Status'
'Avg In %Util'

Order by: order by 'Date' desc, 'Router', 'Port'

Figure 11-8: Order Window

To change the sort order, first delete the existing sort order by clicking on the Clear button. Next select whether the keys (i.e. table columns) should be sorted in ascending or descending order, by clicking on the appropriate button in the Order box. Select the first sort field from the Key box by clicking on the field name in the box. The key and order you choose will appear on the Order by line. You can choose multiple sort keys, with a different order for each key.

The sort will occur in the order you select the fields.

- Click the OK button to have the sort order you specified take effect.
- Click the Reset button to return to the default sort order.
- Click the Cancel button to quit out of the window without changing the current sort order.

Formatting Columns in the Report

You can alter the order of the columns in the report, as well as the precision of numeric columns, and the width of each column, to make the report more readable.

Column Order

You can reorder the columns in the report by dragging and dropping them into different positions.

Grab a column by clicking on its column heading (the cell that contains the column name) and holding down the select key on the mouse. The box around the cell will be displayed as a broken line. Continue to hold the mouse button down while moving the mouse. The broken lined box will move with the cursor on your screen. Move the cursor on top of another column heading, and release the mouse button. The column you are moving will be placed to the right of the column on top of which you released the broken-lined box.

If the report is wider than the TRENDsheet window and you want to move a column to a position not displaying on the screen, move the column to the last position in the window, scroll the report over in direction you are moving the column, and move the column again.

Column Order and Report Sort Order

You can make your report much easier to understand if you put the left-hand columns in the report in the same order as your sort order. For example, if you are sorting a report on line utilization by utilization level and time of day (Received Time), put the Utilization column in the left-hand column of the report and Received Time in the column to the right of Utilization by dragging each column to the left of the fixed Time Period column.

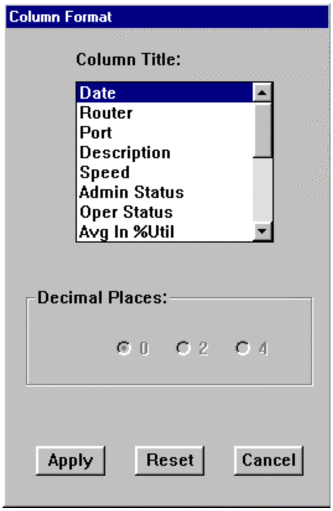
Note that you cannot move a fixed column (see “Fixed Columns” on page 11-16). To put a different column's data in the first column, drag that column to the left of the fixed column.

Fixed Columns

Fixed columns are data columns at the left-most side of the report which cannot be moved to new positions nor be replaced by other columns. They also continue to display in the window if you scroll the report to look at different columns.

Numeric Column Precision

By default, numeric columns are displayed with four decimal places of precision. You can change this using the Column Format window. Bring up the window by selecting the Column Format option under the View button:


Column Format

Column Title:

- Date
- Router
- Port
- Description
- Speed
- Admin Status
- Oper Status
- Avg In %Util

Decimal Places: ● 0 ○ 2 ○ 4

Figure 11-9: Column Format Window

Select the column whose precision you want to change by clicking on its name in the **Column Title** box. Click the number of **Decimal Places** you want the column to display in the report. Values are rounded to the precision you choose. If you select a non-numeric column, the values in the **Decimal Places** box will be grayed out.

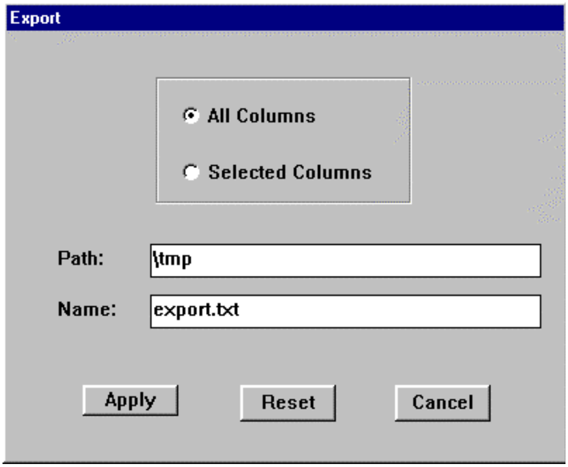
- Click the **Appl**y button to the set the new precisions on the columns modified.
- Click the **Reset** button to set all values back to their defaults.
- Click the **Cancel** button to quit out of the window.

Column Width

To make a column wider or narrower (e.g., after changing the precision of a numeric column), click the right edge column divider of the cell header and hold the mouse key down. The column divider will become a broken line. While still holding the mouse button down, drag the broken line to the left or right to make the column narrower or wider respectively. When the column is the desired width, let go of the mouse button and the column will resize itself to the new width. Changing the width of one column will not affect the width of the other columns.

Exporting Data

You can write the data shown in the TRENDsheet report to an ASCII file on disk. To export the data in your report to an ASCII text file on disk, select the Export option under the File menu. This will bring up the Export window:



● All Columns
○ Selected Columns

Path: \tmp

Name: export.txt

Figure 11-10: Export Window

If every column in the report will be written to the file, click the All Columns box. If you only want specific columns exported, click the Selected Columns box. Select the columns you want exported by double clicking on their column heading cells in the report window. The selected columns will be highlighted.

Enter the directory path where you want the file written on the Path line. The default is your home directory. Enter the name of the file to be written on the Name line. The default is `export.txt.`

- Click the Apply button to export the data from your report to the file. The Apply button will remain grayed out while the export is running.
- Click the Reset button to change the values in the window back to what they were the last time you did an Apply or to their default values if you have not clicked on the Apply button since bringing up the window.
- Click the Cancel button to quit out of the Export window.

Format of a TRENDsheet Export File

An `export file` contains one line for each row in the report. The fields are written to the file in the same order in which they appear in the report. The fields are separated by tab characters. The first line in the report lists the names of the fields in the file. The names, which can contain embedded spaces, are also separated by tab characters. Here is sample data from a TRENDsheet report as it looks in an export file:

```
Received TS - Date    Received TS - Time    Packets    Broadcasts

Multicasts            Collisions            Octets

10-17-94              08:15:00              1122       1003

84                    0                     559808

10-17-94              08:10:00              1134       1009

87                    0                     565696

10-17-94              08:05:00              1078       959

85                    0                     528192

10-17-94              08:00:00              1158       1026
```

```
91                  0                   567104
10-17-94            07:55:00            1195      1069
92                  0                   597760
10-17-94            07:50:00            1126      1007
84                  0                   560576
10-17-94            07:45:00            1019      897
88                  0                   496704
10-17-94            07:40:00            1134      1007
87                  0                   563200
10-17-94            07:35:00            1205      1082
88                  0                   598336
10-17-94            07:30:00            1130      1006
90                  0                   563136
```

If you export a field with a data type of datetime, such as the Received TS column, its values are divided into two fields - a date field with the format MM-DD-YY, and a time field with the format hh:mm:ss. For example, the field called Received TS in the report shown in the TRENDsheet window, is written to the export file as the fields Received TS - Date and Received TS - Time. Numeric fields are written to the file using the same precision with which they are displayed in the report.

If you look at the export file, it may seem that each row from the table created more than one line in the file. This just means the lines are wider than your window and have wrapped to fit in the available space.

Command Line Options

TRENDsheet is invocable from a command line. Enter the command **aga_sheet -h** for a description of the available command line options.

aga_sheet	[**-b** <database>]
	[**-c** <printer copies>]
	[**-e** <db end time in Sybase format>]
	[**-f** <spreadsheet>]
	[**-h**]
	[**-i** <print file>]
	[**-k** <key id from key table>]
	[**-p** <printer>]
	[**-s** <db start time in Sybase format>]

The Sybase date and time format is `‘yyyy-mm-dd hh:mm:ss’`. For example:

```
‘1997-06-10 00:00:00’
```

If you do not specify an extension for the <print file> file name, the output is saved as a postscript (.ps) file. You can specify an extension of .bmp or .gif to save the output in one of these formats instead.

TREND

12 TRENDgraph

TREND utilizes time-related data that can be viewed in sequence to identify usage patterns. TRENDgraph provides easy to use, powerful graphing functions that allow you to define, display, and print reports. In creating line, cumulative area, or stacked bar graphs, the user can select data to be displayed, create expressions, define plot line styles and colors, control X- and Y-Axis scaling, assign report titles and axis labels, and plot multiple statistics from single or multiple sources simultaneously (time-synchronized for comparison). Although data are stored in a relational database, the user does not have to know SQL to create or run reports. All functions can be carried out through TREND's windowed interface.

Main Display and Menu System

You invoke TRENDgraph from the Tools menu of the Trend main window. The TRENDgraph window appears initially with a blank display area and a menu bar providing access to all TRENDgraph functions explained in this chapter (Figure 12-1).

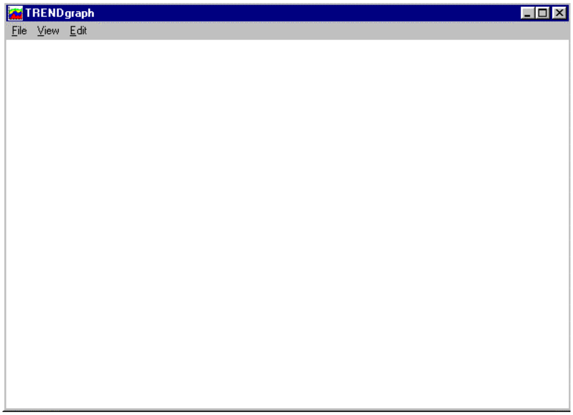
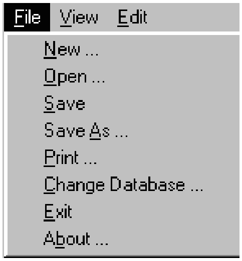


Figure 12-1: TRENDgraph Main Window

Menu Command Summary

File



Use the **New** command to create a new graphical report (see “Creating a New Graph” on page 12-5).

The **Open** command is used to retrieve a pre-defined graphical report (see “Opening an Existing Graphical Report” on page 12-4).

The **Save** option is used to store previously saved reports (see "Selecting Statistics to be Graphed" on page 12-6).

The **Save As** option is used to store new reports or to copy or move existing reports (see "Selecting Statistics to be Graphed" on page 12-6).

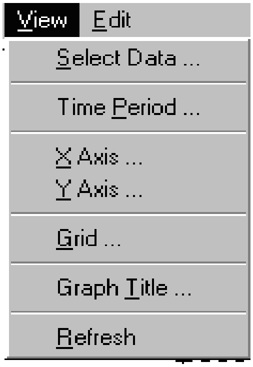
Graphical reports can be printed to a postscript printer or to a file using the **Print** command (see "Print" on page 1-17).

The **Change Database** command provides access to a different SQL server to display its data (see "Displaying Data from a Different Database" on page 12-12).

Use the **Exit** command to quit out of TRENDgraph (see "Selecting Statistics to be Graphed" on page 12-6).

The **About** command invokes TREND version information (see "About TREND" on page 1-19).

View



Use **Select Data** to change the data source node for a report (see "Selecting Data" on page 12-8).

Set the time range of data to be reported on with the **Time Period** option (see "Setting the Time Period to be Displayed" on page 12-13).

Change the time scale with the **X-Axis** command (see “Setting X-Axis Parameters” on page 12-16).

Change value ranges of plotted statistics with the **Y-Axis** command (see “Setting Y-Axis Parameters” on page 12-17).

The **Grid** command provides a toggle for adding or removing grid lines from a report (see “Adding Grid Lines” on page 12-20).

Graph Title allows the addition of meaningful titles to graphical reports (see “Specifying Report Titles” on page 12-19).

Use the **Refresh** command to update the current graph with current data.

Edit

Modify definitions made in an existing reports using the **Aliases/Expressions** command (see “Modifying an Existing Report” on page 12-5).

Opening an Existing Graphical Report

Existing reports are stored as files on disk. You open them by loading them into TRENDgraph. Click the Open option under the File menu. This will bring up the File Selection window (see Figure 1-2: The Select File Window).

Modifying an Existing Report

Any of the defined display parameters can be modified at any time by substituting new values and repeating the steps followed to create the original format. To modify the definitions setup through TRENDbuild, click the **Aliases/Expressions** option under the **Edit** menu. This will put you into TRENDbuild in the context of the current report. That is, the TRENDbuild category, group, statistics, aliases and expressions windows will contain the values used by the report being modified. After you finish making changes in TRENDbuild, reload the report into TRENDgraph using the **Open** option under the **File** menu for those changes to be reflected in the report on your screen.

If you made any changes to the report definition before requesting **Aliases/Expressions,** TRENDgraph will ask if you want to save those changes before proceeding.

Creating a New Report from an Existing Report

There may be times when you want to use one report as the basis for another. To do this, Click the Save As option under the File menu. This will bring up the File Selection window, allowing you to write the current report definition to a new report file on disk. The new file can be modified to create a different report, then saved to preserve the changes.

Creating a New Graph

To create a new graph, select **New** from the **File** menu of the TRENDgraph window. This command invokes the TRENDbuild application where new reports are defined.

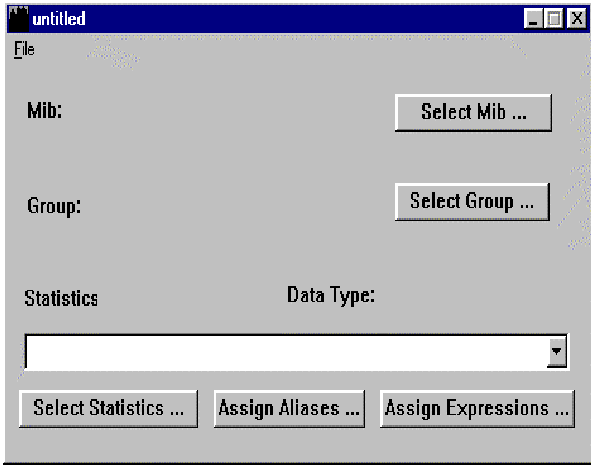


Figure 12-2: TRENDbuild Window

Selecting Statistics to be Graphed

The first step in creating a new graph is to select the category (MIB) and group of the collected data to be reported upon. Use the **Select Mib** and **Select Group** buttons in the Report Scheduler window.

EXAMPLE: *Selecting the Hourly MIB-II ifEntry Group*

To create a report based on hourly mib-II ifEntry data, first click on Select Mib and select mib-II from the Select Mib pop-up box. Click OK.

Next click on Select Group and hour_mib-II_ifEntry from the Select Group pop-up box. Click OK.

The next step is building expressions to be graphed using the available statistics. Click on **Assign Expressions** box in the Report Scheduler window to invoke the **Assign Statistic Aliases** window (Figure 12-3).

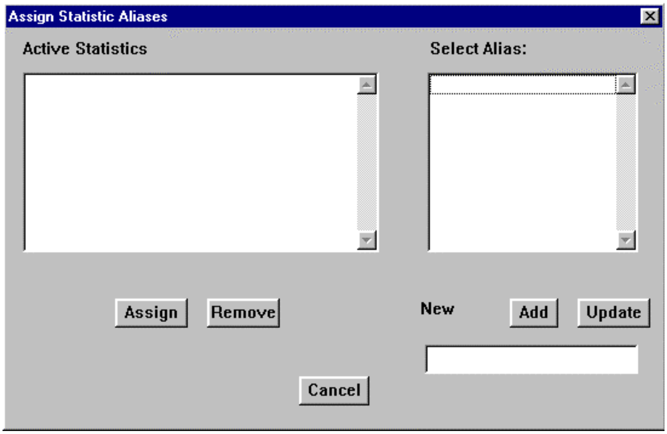


Figure 12-3: Assign Statistic Aliases Window

The **Assign Statistic Aliases** window provides:

- Selection of statistics that will be included in expressions to be graphed (upper left),
- Mathematical operators to be used in expressions (beneath Select Statistics box),
- Creation of expressions from statistics and operators (bottom),
- Naming and adding/deleting expressions, and
- Listing of **Active Expressions** assigned (upper right).

EXAMPLE: *Creating an Input Utilization Report vs. an Output Utilization Report*

To specify input utilization for a device, first click in the New Expressions: box and enter a name for the expressions to be created (e.g., Input_Utilization). Click Add to place the expression in the Active Expressions box. Click the new expressions name in the Active Expressions box to continue (make sure it is highlighted).

Select the following combination of statistics and operators to create the expression for Input Utilization at the bottom of the Assign Statistic Aliases window:

((((ifInOctets*8)/ifSpeed)/delta_time)*100.0)

It is important that the left and right parentheses are paired. Click Add to View to save the expression.

Create an expression for Output Utilization in the New Expressions: box (e.g., Output_Utilization). Click Add. Select Output-Utilization from the Active Expressions box. Enter the following formula in the Output_Utilization input box:

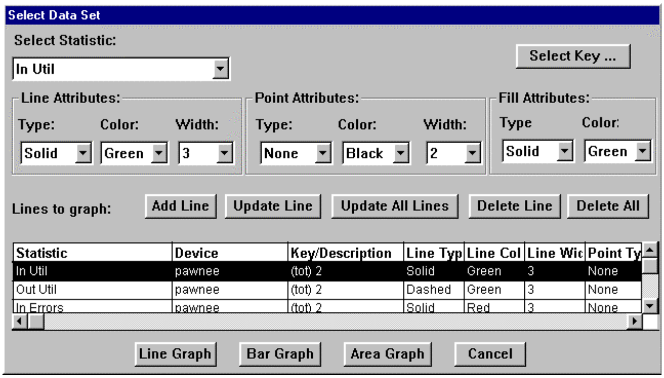
```
((((ifOutOctets*8)/ifSpeed)/delta_time)*100.0)
```

Click on **Add to View** to save the expression.

When all expressions to be included in the graphical report have been created, click **Cancel** to exit the **Assign Statistic Aliases** window. Once you have finished defining the basic report in Report Scheduler, you must save the query file you have created. The Save, Save As, and Exit options on the Report Scheduler File Menu will each invoke the File Selection window, prompting you to store the file under a name of your choice. The file will then be automatically loaded into TRENDgraph. After saving the report, select Exit to quit Report Scheduler. You will automatically return to TRENDgraph.

Selecting Data

For new reports, TRENDgraph will automatically invoke the **Select Data Set** window, allowing you to define plot lines. The Select Data Set window can also be brought up manually by clicking on the Select Data option under the View menu:



Select Statistic: In Util

Line Attributes: Type: Solid Color: Green Width: 3

Point Attributes: Type: None Color: Black Width: 2

Fill Attributes: Type Solid Color: Green

Statistic	Device	Key/Description	Line Typ	Line Col	Line Wic	Point Ty
In Util	pawnee	(tot) 2	Solid	Green	3	None
Out Util	pawnee	(tot) 2	Dashed	Green	3	None
In Errors	pawnee	(tot) 2	Solid	Red	3	None

Figure 12-4: Select Data Set Window

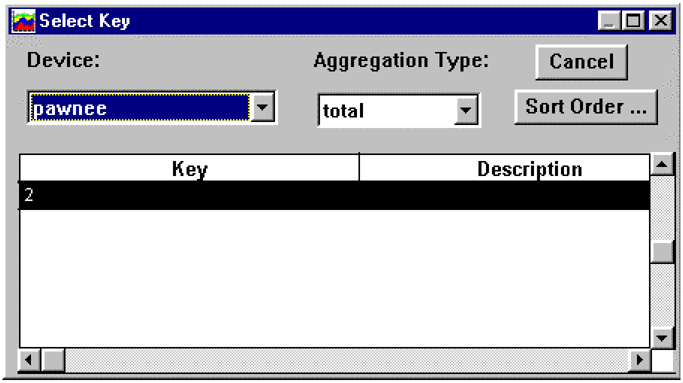
Each plot line displays the values of a statistic or expression for a particular device and key over time. Type, color and width of lines and points can be modified here. As new lines are defined, their definitions are displayed in the Lines to graph box in the Select Data Set window. The existing line definitions can be modified or deleted.

Note: The etherHistoryEntry field 'etherHistoryUtilization' is recorded in 1/100ths of a percent. If utilization is 37.25%, it is displayed as 3725 in TRENDgraph.

Selecting Statistics

The Select Statistic box contains a list of statistics chosen and expressions created, in Report Scheduler when the report was created. Select an expression using the Select Statistic box's down arrow button, then clicking on the desired statistic. Use the scroll bar to see all the available names.

The Select Key box contains a list of all keys for the selected device in the database table being reported on by the graph. Click the Device box down arrow to view a list of devices and select a device for reporting. Select an aggregation type using the Aggregation Type pull down arrow to view and select it.



Device: pawnee

Aggregation Type: total

Key	Description
2	

Figure 12-5: Select Key Window

Select a key by scrolling down and clicking on the appropriately numbered row. Use the scroll bar to see all the available keys. Note that this mechanism allows you to look at the same data from two different devices, or for two keys on the same device simultaneously. TRENDgraph aligns all data by time, so it is easy to compare concurrent values from two sources in a single graph.

In the Select Data Set window, click on **Add Line.** Continue to select keys and aggregation types and add lines until all lines have been specified for the graph.

Note: The Select Key window must be open to ADD or UPDATE lines in the Select Data Set window.

EXAMPLE: *Selecting Data for an Average Input vs. Output Utilization Report*

Using the pull down arrow in the Select Statistic: box, recall the Input_Utilization expression created previously created in the Assign Statistic Aliases window. Click on Select Key to invoke the Select Key window.

Use the pull down arrow in the Device: box to highlight the node from which you collected data. Select a Key by clicking on the desired row in the Key/Description table.

Select average from the Aggregation Type pull down. Click on Add Line in the Select Data Set window.

Repeat the above steps for Output_Utilization. First select Output_Utilization using the Select Statistic pull down arrow in the Select Data Set window.

Displaying the Graph

A basic graph can now be displayed. Line, bar, and area formats are available, and should be selected according to their ability to convey the information in the report. At the bottom of the Select Data Set window, point-and-Click the Line, Bar, or Area graph box. You can choose to define graph attributes before or after displaying a report for the first time by following the paragraphs below.

Note: Bar and area graphs present cumulative displays of data. Time periods must intersect.

Selecting Different Data

Displaying Data for a Different Node

When a TRENDgraph report is created, plot lines must be defined. Statistics or expressions must be identified as well as which device/key pair is to be the source of the data. To change the data source node for a report, select the Select Data option under the View menu. This will bring up the Select Data Set window (Figure 12-4). Follow these steps for each plot line:

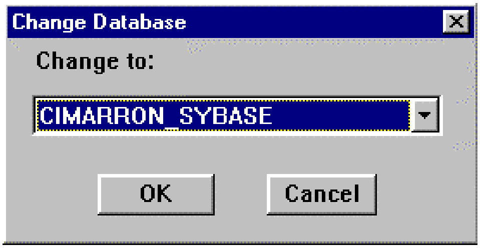
1. Click on a row in the Lines to graph box. The window will highlight the definition for that line.
2. Set the desired values for the Select Device and Select Key boxes.
3. Click the Update Line button.

When the lines are set the way you want, Click the Graph button to display the data from the new source. Note that you select the device/key pair for each plot line independently. You can have different nodes providing data to the same report. You do not have to use a single node as the source for all data.

You may want to save the report again so the next time it runs it uses the current data source.

Displaying Data from a Different Database

You can access a different SQL Server to display its version of the information presented in this report. Click the Change Database option under the File menu to bring up the Change Database window:



Change to: CIMARRON_SYBASE

Figure 12-6: Change Database Window

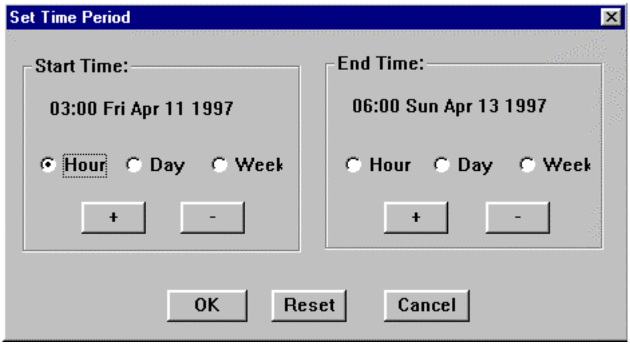
Note: A report may not display the same information for both databases.

The name of your current database is displayed in the Change to box. To change to a different server, Click the down arrow in the Change to box to see a list of the Sybase SQL Servers known to this machine. The values presented here are read from the `$SYBASE/interfaces` file. Select the name of the server you want to access in the report.

- Click the OK button to change to the selected database.
- Click on Cancel to quit out of the window without changing servers.

Setting the Time Period to be Displayed

TRENDgraph allows you to set a Start Time and an End Time for data to be displayed in reports. If you select Time Period from the View Menu, the Set Time Period window is displayed.



Start Time: 03:00 Fri Apr 11 1997

End Time: 06:00 Sun Apr 13 1997

Figure 12-7: Set Time Period Window

Change the Start and End times by selecting the button that corresponds to the portion of the time you wish to set (i.e., Hour, Day, or Week). Next, click on either the + or - buttons to increase or decrease this value respectively.

- Click the OK button to have the sort order you specified take effect.
- Click the Reset button to return to the default start and stop times.
- Click the Cancel button to quit out of the window without changing the start and stop times.

Setting Report Display Parameters

To complete the definition of the new report, you must set its display parameters, including defining the plot line types, setting the X- and Y-Axis parameters, and specifying report titles. These modifications are handled through the Select Data Set window (Figure 12-4).

Changing Line Attributes

Line type, color and width can be specified for each line plotted on a line graph.

The **Type** box is a list of available line patterns. Select the pattern for the current line by clicking on the Type box's down arrow button, then clicking on the desired type. Use the scroll bar to view the list of available patterns.

The **Color** box is a list of the available line colors. Select the color for the current line by clicking on the Color box's down arrow button, then clicking on the desired color. Use the scroll bar to view the list of available colors.

The **Width** box is a list of the available line widths in points. Select the width for the current line by clicking on the Width box's down arrow button, then clicking on the desired width. Use the scroll bar to view the list of available line widths.

Note: Keep in mind how the graph will look on paper while you set it up, especially if you have a color screen and a monochrome printer. Besides the shades of gray available, only line type and width will be distinguishable on a monochrome printout.

Changing Point Attributes

On a line graph, points show where actual data points fall on the plot line. They make it easy to see how far apart your data samples occur.

The **Type** box is a list of the available point types (e.g., dot or box). Select the point type for the current line by clicking on the Type box's down arrow button, then clicking on the desired type. Select **None** if you do not want points included on the current line.

The **Color** box is a list of the available point colors. Select the point color for the current line by clicking on the Color box's down arrow button, then clicking on the desired color. Use the scroll bar to see all the available colors.

The **Width** box is a list of the available point widths in points. Select the point width for the current line by clicking on the Width box's down arrow button, then clicking on the desired width. Use the scroll bar to see all the available point widths.

Defining Fill Attributes

Area and bar chart fill patterns and colors can be changed in the **Fill Attributes** area of the Select Data Set window.

Select the fill pattern by clicking on the pull down arrow on the **Type** box, then clicking on the desired fill type.

Similarly, fill colors can be changed using the **Color** box.

Adding, Changing and Deleting Plot Lines

To add a line definition to the graph once its parameters are set, click the **Add Line** button. The new line definition will appear in the **Lines to graph** box.

To change the definition of an existing plot line, Click the line to be changed in the **Lines to graph** box. Its parameters will be displayed in the appropriate areas in the window. Make the desired changes, then click the **Update Line** button. The line's definition in the **Lines to graph** box will display the new values. Use the **Update All Lines** button to update the values of all lines after making changes to multiple lines.

To delete a line, Click the desired line in the **Lines to graph** box, then Click the **Delete Line** button.

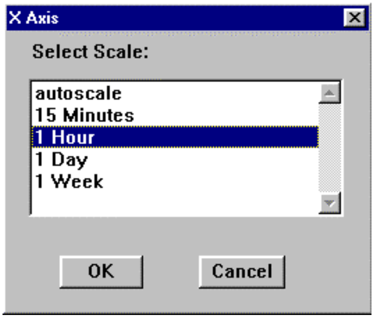
To delete all the lines in the graph, Click the **Delete All** button.

When you want to display the defined lines in the TRENDgraph window, Click the **Graph** button. The graph will redraw according to the current line definitions.

Setting X-Axis Parameters

The horizontal, or X-axis represents time in TREND graphs. You can adjust the time scale to affect how spread out or compact the graph will be. The larger the time scale increment, the more information will appear in a smaller space. The trade-off is crowded data. The default setting is **Autoscale.**

To change the X-axis time scale, select the X-Axis option under the View button to bring up the X-Axis window:



X Axis

Select Scale:

autoscale
15 Minutes
1 Hour
1 Day
1 Week

Figure 12-8: X Axis Window

The time scale options, in the Select Scale box include:

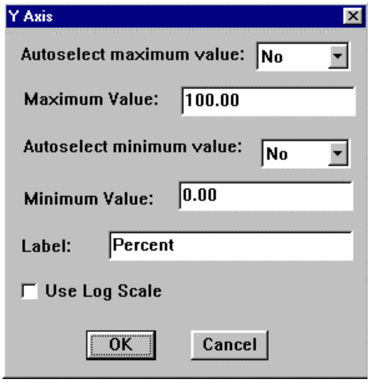
autoscale	Scale graph to the optimal scale automatically so all available data fits in a single window.
15 Minutes	Each hash mark on the X-Axis represents 15 minutes.
1 Hour	Each hash mark on the X-Axis represents 1 hour.
1 Day	Each hash mark on the X-Axis represents 1 day.
1 Week	Each hash mark on the X-Axis represents 1 week.

- Click the **OK** button to activate the new scale.
- Click on **Cancel** to retain the original scale.

Setting Y-Axis Parameters

The vertical, or Y-Axis represents the values of the statistics being plotted. You can control whether the axis automatically scales itself to exactly fit the actual range of values displayed or locks in a particular maximum and minimum value and excludes anything that exceeds this range. TREND selects auto scaling by default. No default axis label is provided.

To change the Y-Axis parameters, select the **Y-Axis** option under the **View** button to bring up the Y-Axis window:



Autoselect maximum value: No

Maximum Value: 100.00

Autoselect minimum value: No

Minimum Value: 0.00

Label: Percent

□ Use Log Scale

Figure 12-9: Y Axis Window

The window displays the current minimum and maximum Y-Axis values from among all plot lines.

Note: The minimum is not necessarily 0.

To change auto-scaling to the maximum value, Click the down arrow of the **Autoselect maximum value** box, and select **Yes** to enable auto-scaling, or No to use an absolute value. If you select No, enter the desired maximum Y-Axis value on the **Maximum Value** line or leave the current value shown.

To change auto-scaling to the minimum value, Click the down arrow of the **Autoselect minimum value** box, and select **Yes** to enable auto-scaling, or **No** to use an absolute value. If you select **No,** enter the desired minimum Y-Axis value on the **Maximum Value** line or you can use the current value shown.

Enter a label for the Y-Axis on the **Label** line. Any printable characters are allowed. The label will be displayed at the top of the Y-Axis.

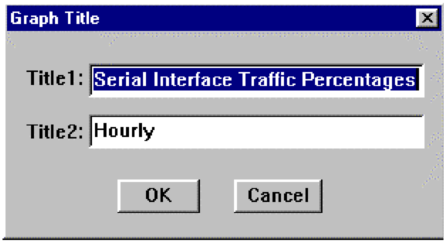
- Click the OK button to activate the new scale.
- Click on Cancel to retain the original scale.

If the range of values on the Y-Axis differ by several orders of magnitude, you may want to use a log scale when displaying a graph. Highlight the **Use Log Scale** box to enact this functionality.

Specifying Report Titles

Report titles are centered at the top of the graph area. Of the three title lines, two can be set by the user while the third is the date range for data presented in the graph.

To change the report titles, select the Graph Title option under the View button to bring up the Graph Title window:



Title1: Serial Interface Traffic Percentages

Title2: Hourly

Figure 12-10: Graph Title Window

The window displays the current report title values. By default, **Title1** is the name of the report file (without the .qgr extension).

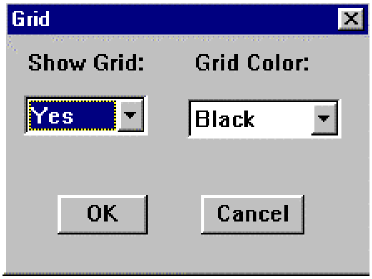
Change the titles by typing in new values.

- Click the OK button to activate the new titles.
- Click on Cancel to quit out of the window without changing the original values.

Adding Grid Lines

Grid lines extend the Y-Axis hash marks across the graph area to make it easier to determine the actual values of data points in the graph. The user can turn grid lines on and off, and change their color. Grid lines will appear in a printout of the report. Keep in mind how they will look on paper while you set them up, especially if you have a color screen and a monochrome printer.

To change the grid line settings, select the **Grid** option under the **View** menu. This will bring up the Grid window:



Show Grid: Yes

Grid Color: Black

Figure 12-11: Grid Window

The window shows the current grid display settings.

To turn grid lines *on* or *off*, Click the down arrow of the **Show Grid** box, and select **Yes** to turn them on, or **No** to turn them off.

To change the color of the grid lines, Click the down arrow of the **Grid Color** box, and select the color you want. Of course the color will only matter if you are displaying *grid lines*.

- Click the **OK** button to activate the revised selections.
- Click on **Cancel** to retain the original values.

Drilling Down for More Detailed Data

TREND graphical reports contain a drill-down feature that allows the user to click on a region of a graphical report to obtain a second, more detailed tabular report displaying the data of interest. This data can point to other reports that will provide a complete picture of the area of concern.

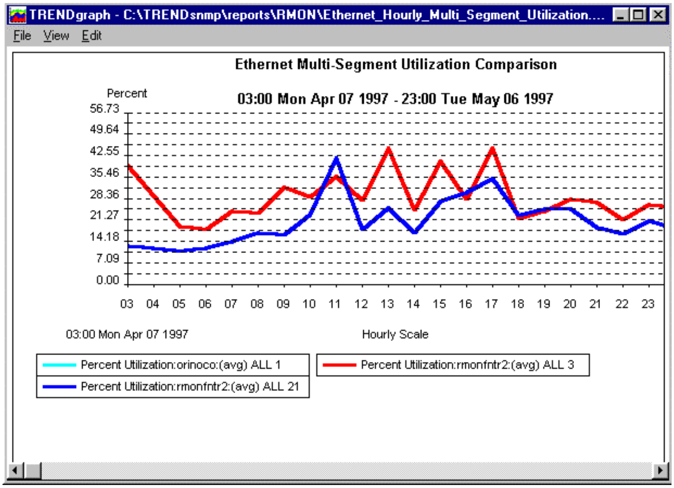

TRENDgraph - C:\TRENDsnmp\reports\RMON\Ethernet_Hourly_Multi_Segment_Utilization....
File View Edit
Ethernet Multi-Segment Utilization Comparison
Percent
03:00 Mon Apr 07 1997 - 23:00 Tue May 06 1997
56.73
49.64
42.55
35.46
28.36
21.27
14.18
7.09
0.00
03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
03:00 Mon Apr 07 1997
Hourly Scale
Percent Utilization:orinoco:(avg) ALL 1
Percent Utilization:rmonfntr2:(avg) ALL 3
Percent Utilization:rmonfntr2:(avg) ALL 21

Figure 12-12: Graphical Report with Supporting Drill-Down Tabular Report

Command Line Options

TRENDgraph is invocable from a command line via the command **agx_graph.** Enter the command **agx_graph -h** for a description of TRENDgraph command options.

agx_graph	[**-b** database]
	[**-c** <printer copies>]
	[**-e** <db end time in Sybase format>]
	[**-f** graph-file]
	[-**h**]
	-i <print file>
	[**-k** <keyid1:keyid2:keyid3: ...>]
	[**-p** <printer>]
	[**-s** <db start time in Sybase format>]

The Sybase date and time format is `‘yyyy-mm-dd hh:mm:ss’`. For example:

```
‘1997-06-10 00:00:00’
```

If you do not specify an extension for the <print file> file name, the output is saved as a postscript (.ps) file. You can specify an extension of .bmp or .gif to save the output in one of these formats instead.

TREND

13 Grade of Service

Grade of Service (GOS) is a tool for system health reporting and Service Level Agreements (SLAs). GOS converts sets of individual historical performance statistics into meaningful graphical displays using a weighted stacked bar format.

The user can select any set of SNMP or RMON metrics and assign a grading scale to each based on the relative importance of that metric. A grading scale (e.g., excellent, good, fair and poor) allows the user to concentrate troubleshooting efforts immediately on areas where performance is substandard.

Grade of Service reports alleviate data overload that can result from massive amounts of network data available from countless devices and sources. Critical parameters for a given device are rolled up into an overall indicator that reflects that device’s performance level.

Main Display and Menu System

Select the Grade of Service button on the TREND Main window to invoke the Grade of Service application.

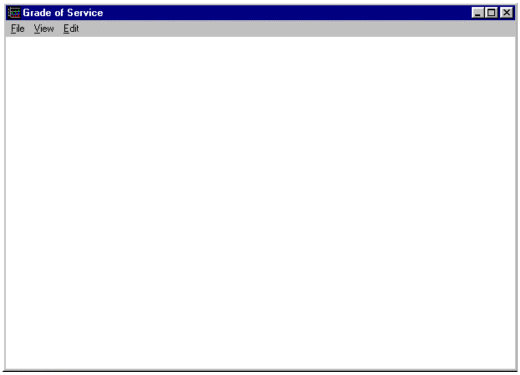
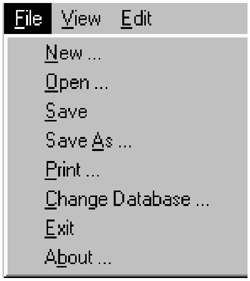


Figure 13-1: Grade of Service Report Window

Command Menu Summary

File



The **New** command is used to create a new Grade of Service report (see "Creating a New Grade of Service Report" on page 13-7).

Use the **Open** command to retrieve a pre-defined GOS report (see "Opening an Existing GOS Report" on page 13-6 and "The Open Command" on page 1-15).

The **Save** option stores previously saved GOS reports.

Save As stores new reports or copies existing reports.

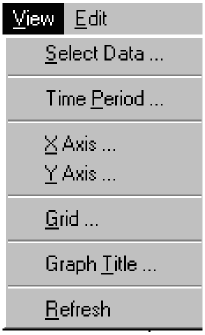
Graphs can be printed directly to a postscript printer by selecting the **Print** command (see "Print" on page 1-17).

The **Change Database** command provides access to data tables in other databases (see Section "Displaying Data from a Different Database" on page 13-14).

Exit quits the Grade of Service application (see "Creating a New Grade of Service Report" on page 13-7 and Section "Exit" on page 1-19.

About provides TREND version information (see "About TREND" on page 1-19).

View



To change the data source node for a report, choose the **Select Data** option. This will bring up the Select Data Set window (see Figure 13-4).

To set the time range of data to be reported, click the **Time Period** option. This will bring up the Set Time Period window (see Figure 13-8, “Set Time Period Window,” on page 13-15).

To change the time scale on the X-axis, select the **X-Axis** option (see “Setting X-Axis Parameters” on page 13-16).

To assign labels (e.g., excellent, good, fair, poor) to the threshold values assigned in the Select Data window, select the **Y-Axis** option (see “Setting Y-Axis Parameters” on page 13-18).

The **Grid** command allows you to toggle the grid on or off and alter grid characteristics used for easy viewing of TREND graphs (see “Adding Grid Lines” on page 13-19).

The **Graph Title** allows you to specify meaningful graph titles and subtitles (see “Specifying Report Titles” on page 13-19).

The **Refresh** option allows the user to rerun the query that generated the graph, for up-to-the-minute data. The graph will clear and redraw itself with the most current data available.

Edit

The **Aliases/Expressions** option in the Edit menu, allows you to modify definitions setup through TRENDbuild (see “Assigning Aliases and Creating Expressions” on page 10-10).

Note: If an alias is set and plotted in GOS and then changed in TRENDbuild, the grapher will still show the original alias because TRENDbuild does not automatically update the lines added to the report file by GOS. You must resave and reopen the file after modifications have been completed.

The Grade-of-Service Concept

The following example explains the basic principles of Grade of Service reporting.

EXAMPLE:

Assume that you wish to compute the grade of service for a particular LAN segment. You determine that the significant parameters contributing to your LAN GOS are Utilization, Collisions, Errors, and Overhead, with relative contributions as follows:

Parameter	**Weighting**
Utilization	40%
Collisions	25%
Errors	25%
Overhead	10%

You next decide that you will include four categories of service in your GOS report: Excellent, Good, Fair, and Poor. In this case, a score of '1' for an individual parameter or weighted, combined parameters will fall in the excellent range. Similarly, scores of '2' will denote good, '3' will be fair, and '4' will be considered poor performance.

You must also determine what ranges of scores for individual parameters constitute excellent, good, fair, or poor service:

Parameter	**Excellent**	**Good**	**Fair**	**Poor**
Utilization	<5%	6%-15%	16%-30%	>30%
Collisions	<5%	6%-10%	11%-15%	>15%
Errors	<1%	2%-5%	6%-10%	>10%
Overhead	<5%	6%-10%	11%-15%	>15%

If, at a given time, you receive a Utilization value of 6% (Good=2), Collisions of 4% (Excellent=1), Errors of 3% (Good=2), and Overhead of 13% (Fair=3); your GOS for that time period is calculated as:

(0.40*2) + (0.25 * 1) + (0.25*2) + (0.10*3)

0.80 + 0.25 + 0.50 + 0.30 = 1.85

In this case, GOS is between '1' and '2', which falls in the 'Good' range.

Opening an Existing GOS Report

Existing reports are stored as files on disk. They are opened by loading them into the Grade of Service application. Click the Open option under the File menu. This will bring up the File Selection window (see Figure 1-2, “The Select File Window,” on page 1-15).

Creating a New Grade of Service Report

The definition of a new report (including selection of the mib) is done in TRENDbuild (See "TRENDsheet" on page 11-1). You can invoke TRENDbuild from inside GOS by selecting the **New** option under the **File** menu.

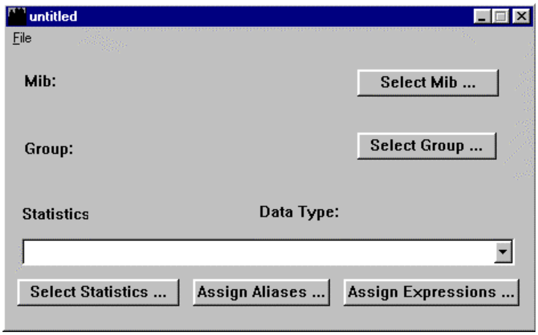


Figure 13-2: TRENDbuild Main Window

Selecting Statistics

The first step in creating a new graph is to select the category (MIB) and group of the collected data to be reported upon. Use the **Select Mib** and **Select Group** buttons in the TRENDbuild window.

EXAMPLE: *Selecting the Hourly MIB-II ifEntry Group*

To create a report based on hourly mib-II ifEntry data, first click **Select Mib** and select *mib-II* from the Select Mib pop-up box. Click OK.

Next click **Select Group** and *hour_mib-II_ifEntry* from the Select Group pop-up box. Click OK.

The next step is building expressions to be graphed using the available statistics. Click the **Assign Expressions** box in the TRENDbuild window to invoke the **Edit Expressions** window (Figure 13-3).

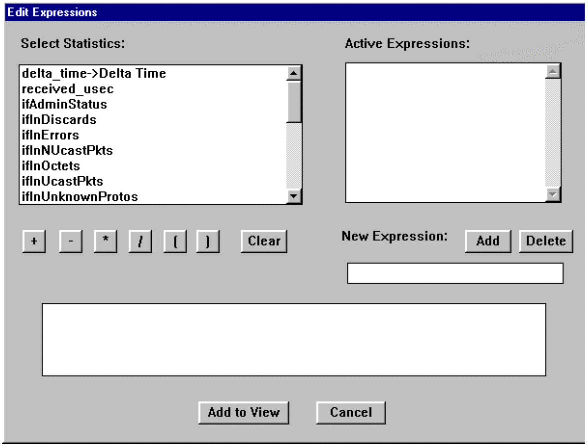


Figure 13-3: Edit Expressions Window

The **Edit Expressions** window provides:

- Selection of statistics that will be included in expressions to be graphed (upper left),
- Mathematical operators to be used in expressions (beneath Select Statistics box),
- Creation of expressions from statistics and operators (bottom),
- Naming and adding/deleting expressions, and
- Listing of **Active Expressions** assigned (upper right).

EXAMPLE: *Creating an Expression: Input Utilization*

To specify input utilization for a device, first click in the **New Expressions:** box and enter a name for the expressions to be created (e.g., Input_Utilization). Click **Add** to place the expression in the **Active Expressions** box. Click the new expressions name in the **Active Expressions** box to continue (make sure it is highlighted).

Select the following combination of statistics and operators to create the expression for Input Utilization at the bottom of the **Edit Expressions** window:

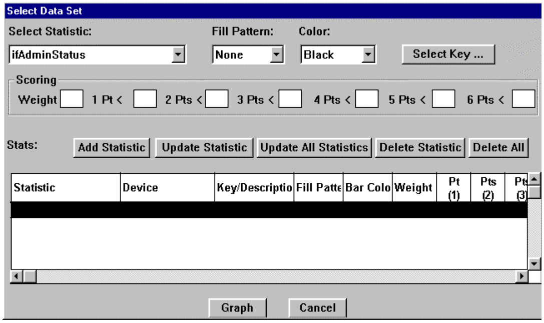
```
    (((ifInOctets*8)/ifSpeed)*100.0)
```

It is important that the left and right parentheses are paired. Click **Add to View** to save the expression.

Once you finish defining the basic report (as described below) in TRENDbuild, you will need to save the query file you have just created. The **Save, Save As,** and **Exit** options on the TRENDbuild **File** Menu will all prompt you with the File Selection window to store the file under a name of your choice. The file will then be automatically loaded into GOS.

Selecting Data

When you run a new report, GOS will bring up the Select Data Set window after you exit TRENDbuild. Alternatively, the Select Data Set window can be brought up by clicking on the **Select Data** option under the **View** menu:



Select Data Set

Select Statistic: ifAdminStatus

Fill Pattern: None

Color: Black

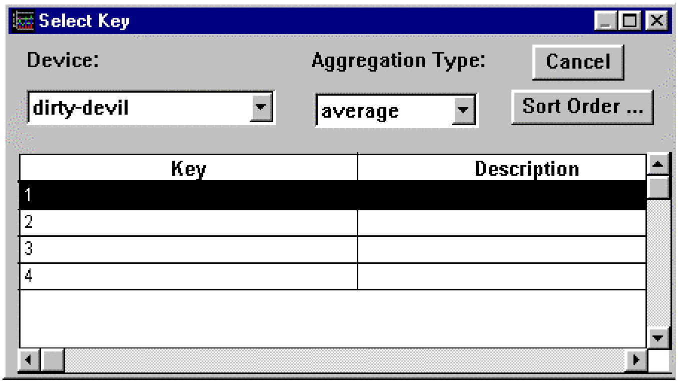
Statistic	Device	Key/Descriptio	Fill Patte	Bar Colo	Weight	Pt (1)	Pts (2)	Pts (3)

Figure 13-4: Select Data Set Window

Note: This window allows the user to establish the relative weight that each statistic selected contributes to grade of service for a given device. The total weight of all statistics selected must be 100.

Total points accumulated from all statistics collected on a device provide a score that reflects that device's level of performance. Points are assigned to each statistic as defined in the table at the bottom of the select data set window. Each statistic contributes points to the overall grade of service according to thresholds that the user defines for that statistic in the Scoring area of the Select Data Set window.

The **Select Key** box invokes the Select Key window (Figure 13-5), which contains a list of all keys for the selected device in the database table being reported on by the graph. Click the Device box down arrow to view a list of devices and select a device for reporting. Select an aggregation type using the Aggregation Type pull down arrow to view and select it.



Device: dirty-devil

Aggregation Type: average

Key	Description
1	
2	
3	
4	

Figure 13-5: Select Key Window

Select a key by scrolling down and clicking on the appropriately numbered row. Use the scroll bar to see all the available keys.

In the Select Data Set window, click Add Line. Continue to select keys and aggregation types and add statistics until all statistics have been specified for the graph.

EXAMPLE: *Creating an Interface Health Report*

An Interface Health report is defined with the following parameters and weighting:

Parameter	Weighting	Excellent	Good	Fair	Poor
% Utilization	50%	<25	25-35	36-50	51-75
% Input Errors	10%	<1	1-2	2-3	3-4
% Output Errors	10%	<1	1-2	2-3	3-4
Receive Pkt Discards	15%	<1	1-3	3-5	5-10
Sent Pkt Discards	15%	<1	1-3	3-5	5-10

Having selected the five statistics (parameters) in TRENDbuild, call up the first statistic to be added to the GOS report (e.g., Percent Utilization) in the Select Statistic box of the Grade of Service -- Select Data Set window.

Click Select Key to invoke the Grade of Service -- Select Key window and set the device, key, and aggregation type desired (see "TRENDbuild" on page 10-1 for additional details about TRENDbuild and the Select Key function).

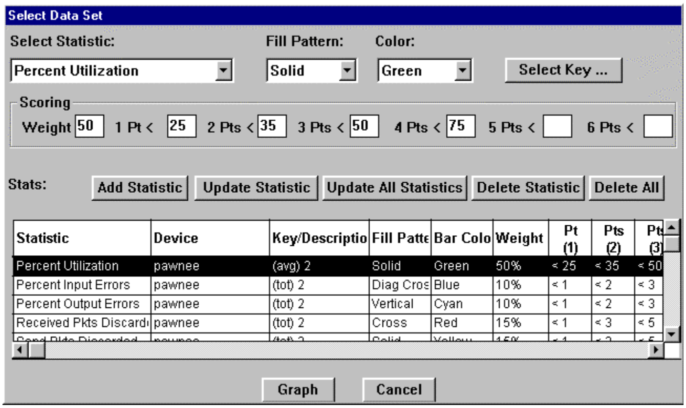
On the View pull-down menu, in the Scoring box of the Grade of Service -- Select Data Set window, enter the relative weight the selected statistic contributes to the health of the interface (e.g., 50 for Percent Utilization).

Enter the upper limit for each scoring level identified next to the number of points that will be assigned to the parameter for a given level of service. For example, for Percent Utilization, one point will be earned for a utilization level under 25%, so 25 must be entered in the box next to 1 Pt. Similarly, 35 will be entered in the 2 Pt box, 50 in the 3 Pt box, and 75 in the 4 Pt box. Keep in mind that a lower number reflects better health in the Grade of Service report.

It is probably easiest to assign a fill pattern and color to represent the current statistic in the GOS stacked bar graph (see "Changing GOS Bar Attributes" on page 13-16 for further details).

Now that the parameter has been fully defined, click Add Statistic to enter Percent Utilization into the **Statistics to graph:** table.

Repeat these steps for the additional four parameters to complete the table, then click Graph at the bottom of the Grade of Service -- Select Data Set window to display the GOS report.



Select Statistic: Percent Utilization

Fill Pattern: Solid

Color: Green

Scoring

Weight 50 1 Pt < 25 2 Pts < 35 3 Pts < 50 4 Pts < 75 5 Pts < 6 Pts <

Statistic	Device	Key/Descriptio	Fill Patte	Bar Colo	Weight	Pt (1)	Pts (2)	Pt (3)
Percent Utilization	pawnee	(avg) 2	Solid	Green	50%	< 25	< 35	< 50
Percent Input Errors	pawnee	(tot) 2	Diag Cros	Blue	10%	< 1	< 2	< 3
Percent Output Errors	pawnee	(tot) 2	Vertical	Cyan	10%	< 1	< 2	< 3
Received Pkts Discard	pawnee	(tot) 2	Cross	Red	15%	< 1	< 3	< 5

Figure 13-6: Hourly Interface Health Report Data Set

Threshold labels are specified for the graph using the Y-Axis command under the View menu (see “Setting Y-Axis Parameters” on page 13-18).

Finally, **Save** the fully defined report.

Selecting Different Data

Displaying Data for a Different Node

When a GOS report is created, statistics or expressions must be identified as well as which device/key pair is to be the source of the data. To change the data source node for a report, select the Grade of Service -- Select Data option under the View menu. This will bring up the Select Data Set window (Figure 13-6). Follow these steps for each statistic:

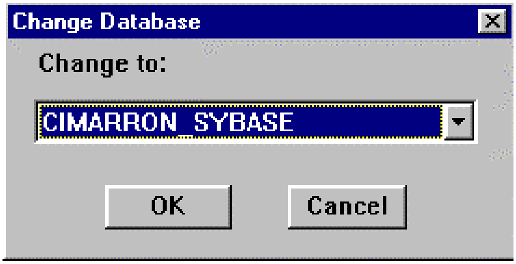
1. Click a row in the **Statistics to graph** box. The window will highlight the definition for that line.
2. Set the desired values for the **Device** and **Key** in the Select Key window.
3. Click the **Update Statistic** button in the Select Data Set window.

When the statistics are set the way you want, click the Graph button to display the data from the new source. Note that you select the device/key pair for each statistic independently. You can have different nodes providing data to the same report. You do not have to use a single node as the source for all data.

You may want to save the report again so the next time it runs it uses the current data source.

Displaying Data from a Different Database

You can access a different SQL Server to display its version of the information presented in this report. Click the Change Database option under the File menu to bring up the Change Database window:



CIMARRON_SYBASE

Figure 13-7: Change Database Window

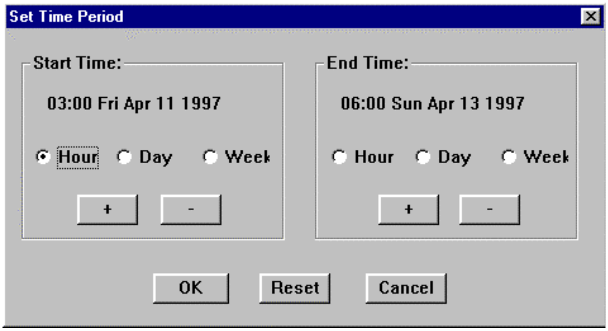
Note: A report may not display the same information for both databases.

The name of your current database is displayed in the Change to box. To change to a different server, click the down arrow in the Change to box to see a list of the Sybase SQL Servers known to this machine. The values presented here are read from the $SYBASE/interfaces file. Select the name of the server you want to access in the report.

- Click the OK button to change to the selected database.
- Click Cancel to quit out of the window without changing servers.

Setting the Time Period to be Displayed

Grade of Service allows you to set a Start Time and an End Time for data to be displayed in reports. If you select Time Period from the View Menu, the Set Time Period window is displayed.



Set Time Period

Start Time:

03:00 Fri Apr 11 1997

● Hour ○ Day ○ Week

End Time:

06:00 Sun Apr 13 1997

○ Hour ○ Day ○ Week

Figure 13-8: Set Time Period Window

Change the Start and End times by selecting the button that corresponds to the portion of the time you wish to set (i.e., **Hour, Day,** or **Week**). Next, click either the + or - buttons to increase or decrease this value respectively.

- Click the OK button to have the sort order you specified take effect.
- Click the Reset button to return to the default start and stop times.
- Click the Cancel button to quit out of the window without changing the start and stop times.

Setting Report Display Parameters

To complete the definition of the new report, you must set its display parameters. This includes defining the plot line types, setting the X- and Y-Axis parameters, and specifying report titles.

Changing GOS Bar Attributes

Color and fill pattern can be specified for each bar segment plotted on a GOS graph.

Select the color for the current statistic by clicking on the Color box's down arrow button, then clicking on the desired color. Use the scroll bar to view the list of available colors.

Select the fill pattern for the current statistic by clicking on the Fill Pattern box's down arrow button, then clicking on the desired pattern. Use the scroll bar to view the list of available patterns.

Note: Keep in mind how the graph will look on paper while you set it up, especially if you have a color screen and a monochrome printer. Besides the shades of gray available, only fill patterns will be distinguishable on a monochrome printout.

Setting X-Axis Parameters

The horizontal, or X-axis represents time in TREND graphs. You can adjust the time scale to affect how spread out or compact the graph will be. The larger the time scale increment, the more information will appear in a smaller space. The trade-off is crowded data. The default setting is **Autoscale.**

To change the X-axis time scale, select the **X-Axis** option under the **View** menu to bring up the X-Axis window:

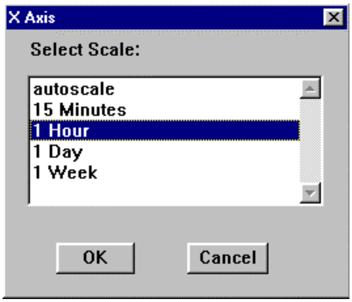


Figure 13-9: X-Axis Window

The time scale options, in the Select Scale box include:

autoscale Scale the graph to the optimal scale automatically so all available data fits in a single window.

15 Minutes Each hash mark on the X-Axis represents 15 minutes.

1 Hour Each hash mark on the X-Axis represents 1 hour.

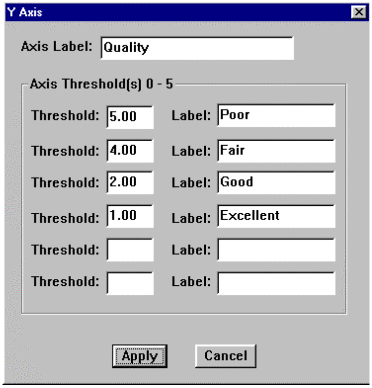
1 Day Each hash mark on the X-Axis represents 1 day.

1 Week Each hash mark on the X-Axis represents 1 week.

- Click the **OK** button to activate the new scale.
- Click **Cancel** to retain the original scale.

Setting Y-Axis Parameters

The vertical, or Y-Axis represents the overall health range based on the statistics plotted. You can assign labels to the threshold values. To add or change the Y-Axis parameters, select the **Y-Axis** option under the **View** menu to bring up the Y-Axis window:



Axis Label: Quality

Axis Threshold(s) 0 - 5

Threshold:	5.00	Label:	Poor
Threshold:	4.00	Label:	Fair
Threshold:	2.00	Label:	Good
Threshold:	1.00	Label:	Excellent
Threshold:		Label:	
Threshold:		Label:	

Figure 13-10: Y-Axis Window

The window displays the current labels assigned to the Y-Axis. To add a new label, click in a blank Threshold box and enter the threshold number corresponding to the points (Pts) in the Statistics To Graph box of the Select Data Set window, which is accessed from the View pull-down menu (see Figure 13-6). Enter the appropriate threshold number and then enter a corresponding label in the *Label box* to the right.

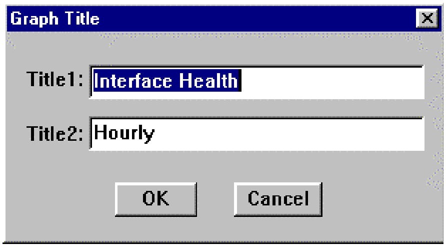
To change a threshold or label, highlight an existing entry and type in the new one.

- Click the **Apply** button to activate the new labels.
- Click **Cancel** to retain the original labels.

Specifying Report Titles

Report titles are centered at the top of the graph area. Of the three title lines, two can be set by the user while the third is the date range for data presented in the graph.

To change the report titles, select the **Graph Title** option under the **View** menu to bring up the Graph Title window:



Title1: Interface Health

Title2: Hourly

Figure 13-11: Graph Title Window

The window displays the current report title values. By default, **Title1** is the name of the report file (without the .qgr extension).

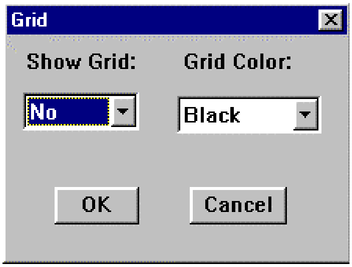
Change the titles by typing in new values.

- Click the Apply button to activate the new titles.
- Click Cancel to quit out of the window without changing the original values.

Adding Grid Lines

Grid lines extend the Y-Axis hash marks across the graph area to make it easier to determine the actual values of data points in the graph. The user can turn grid lines on and off, and change their color. Grid lines will appear in a printout of the report. Keep in mind how they will look on paper while you set them up, especially if you have a color screen and a monochrome printer.

To change the grid line settings, select the **Grid** option under the **View** menu. This will bring up the Grid window:



Show Grid: No

Grid Color: Black

Figure 13-12: Grid Window

The window shows the current grid display settings.

To turn grid lines on or off, click the down arrow of the **Show Grid** box, and select **Yes** to turn them on, or **No** to turn them off.

To change the color of the grid lines, click the down arrow of the **Grid Color** box, and select the color you want. Of course the color will only matter if you are displaying grid lines.

Click the **OK** button to activate the revised selections.

Click **Cancel** to retain the original values.

Drilling Down for More Detailed Data

Grade of Service reports contain a drill-down feature that allows the user to click a region of the graph to obtain a second, detailed tabular report, displaying the data of interest.

Command Line Options

The Grade of Service application is invocable from a command line. the command **agx_gos -h** to see a description of the GOS command line options.

agx_gos	[**-b** database]
	[**-c** <printer copies>]
	[**-e** <db end time in Sybase format>]
	[**-f** graph-file]
	[**-h**]
	-i <print file>
	[**-k** <keyid1:keyid2:keyid3: ...>]
	[**-p** <printer>]
	[**-s** <db start time in Sybase format>]

The Sybase date and time format is `‘yyyy-mm-dd hh:mm:ss’`. For example:

```
‘1997-06-10 00:00:00’
```

If you do not specify an extension for the <print file> file name, the output is saved as a postscript (.ps) file. You can specify an extension of .bmp or .gif to save the output in one of these formats instead.

TREND

14 Report Launcher

The Report Launcher allows you to run a TREND-installed or user-defined report against data from any available database on an ad hoc basis. This chapter describes how to:

- Invoke the Report Launcher.
- Select the database against which the report is run.
- Select the report.
- Select the devices and elements to include.
- Specify the report time period.
- Generate a report.

Invoking the Report Launcher

You invoke the Report Launcher by clicking on the Report Launcher button in the TREND main window:

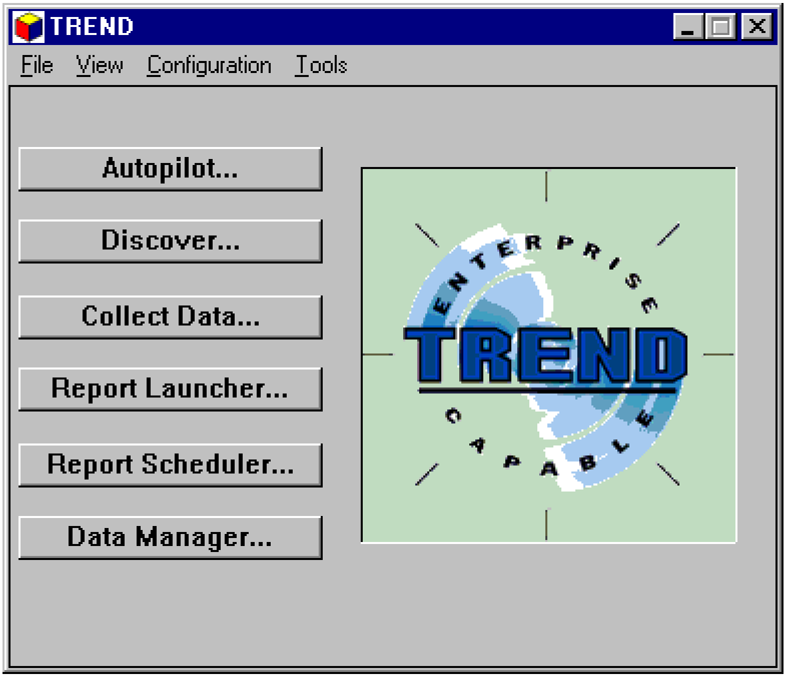
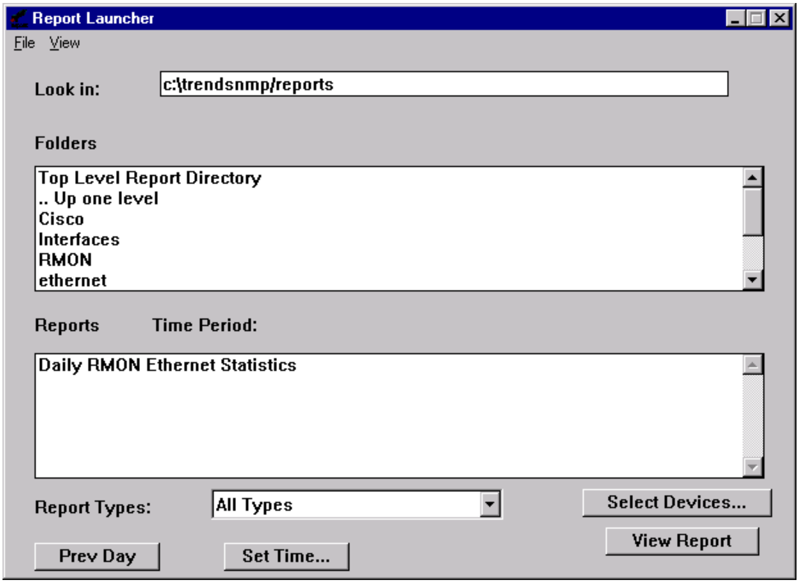


Figure 14-1: TREND Main Window

In response, the Report Launcher window appears:



Look in: c:\trendsnmp/reports

Top Level Report Directory
.. Up one level
Cisco
Interfaces
RMON
ethernet

Time Period:

Daily RMON Ethernet Statistics

Report Types: All Types

Figure 14-2: Report Launcher Window

The controls in the Report Launcher window are:

Control	Description
Look in edit box	Identifies the current report directory. The *current directory* determines the list of reports displayed in the Reports list box.

Control	Description
Folders list box	Lists the Top Level Report Directory and its subdirectories. You can select any report directory to be the Top Level Report Directory. We tell you how to do this in "Selecting the Report File Directory" on page 14-7. The Folders list box makes it easy for you to select a directory to be the current directory. You can easily change the value of the current directory by double-clicking on an entry in the Folders list box.
Reports list box	Lists the reports defined in the current directory. You select the report you want to create from this list box.
Report Types list box	Enables you to restrict the contents of the Reports list box to a specific report type (for example, all types, TRENDgraph, TRENDsheet, and so on).
Prev (Day) button	Provides an easy way to produce a report for the previous reporting period. The label on this button is Prev Hour, Prev Day, Prev Week, or Prev Month depending on whether the selected report is an hourly, daily, weekly, or monthly report, respectively.
Set Time button	Enables you to specify (or change) the report time period.
Select Devices button	Enables you to select specific devices and elements to include in the report.
View Report button	Launches the report.

Remaining sections in this chapter describe how to:

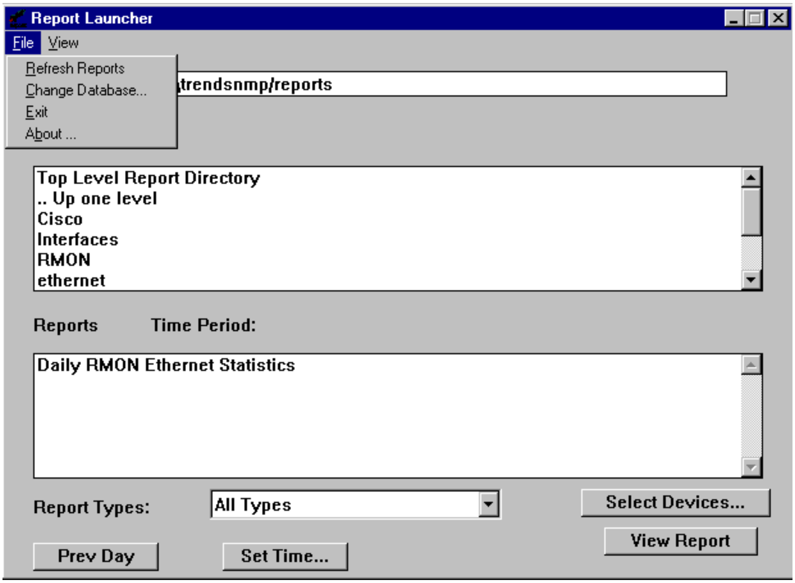
- Select the database to be reported (see "Selecting the Database" on page 14-5).
- Specify the report file directories to be used (see "Selecting the Report File Directory" on page 14-7).

- Select a report type and report (see "Selecting a Report" on page 14-11).
- Select the devices and elements to include (see "Selecting Devices and Elements" on page 14-14).
- Specify the report time period (see "Specifying the Report Time Period" on page 14-18).
- Launch the report (see "Generating the Report" on page 14-20).
- Refresh the report list (see "Refreshing the Report List" on page 14-20).

Selecting the Database

The default database from which report information is taken is identified by the value of the DSQUERY environment variable. Take the following actions to select a different database:

1. Select Change Database from the File menu:



trendsnmp/reports

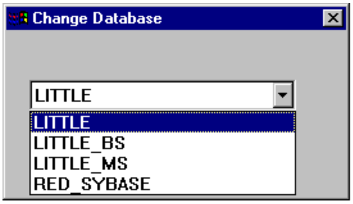
Top Level Report Directory
.. Up one level
Cisco
Interfaces
RMON
ethernet

Daily RMON Ethernet Statistics

Report Types: All Types

Figure 14-3: Report Launcher Window’s File Menu

The Change Database window appears:



LITTLE

LITTLE
LITTLE_BS
LITTLE_MS
RED_SYBASE

Figure 14-4: Change Database Window

2. Select the desired database from the entries in the list box.

Note: Selecting a database from the Change Database window only identifies the database to be used for reporting with Report Launcher. It does not change the value of the DSQUERY environment variable.

Selecting the Report File Directory

TREND-installed and user report definitions are stored in one or more report directories. (The default directory for TREND-installed report definitions is $DPIPE_HOME/reports.)

You can identify the directories you want to work with and make it easy to browse their contents by designating a Top Level Report Directory. When you do this, that directory becomes the initial current directory. If you select another directory to be the current directory, you can easily make the Top Level Directory the current directory again by double-clicking on the Top Level Report Directory entry in the Folders list box.

The name of the Top Level Report Directory stays the same (and survives the termination of the Report Launcher session) until you change it as described below.

Remaining entries in the Folders list box include:

- Names of the first-level subdirectories of the directory that you have designated as the Top Level Report Directory. You can double-click on a subdirectory name to make it the current directory. The name will appear in the Look in edit box. (You can also type the directory name in the Look in edit box.)
- The ..Up one level entry. Double-click on this entry to make the immediate parent of the current directory the new current directory.

The *current directory* is always the one identified in the Look in edit box in the Report Launcher window. The entries in the Reports list box are always taken from the current directory.

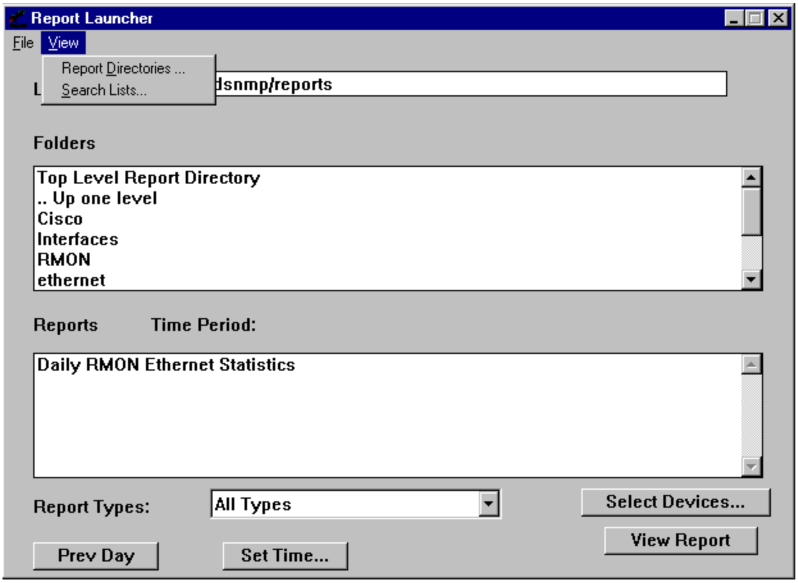
A summary of the types of directory-navigation actions you can take from the folders list box follows (refer to Figure 14-2: Report Launcher Window as you review these actions):

- c:\trendsnmp/reports is the current directory.
- c:\trendsnmp/reports is also the Top Level Report Directory (but you cannot tell this just by looking at the Report Launcher window. The only way you can actually display the current name of the Top Level Report Directory is to double-click on the Top Level Report Directory entry in the Folders list box. The name of that directory will appear in the Look in edit box.)
- Double-clicking on the Up one level entry will make c:/trendsnmp the current directory.
- Cisco, Interfaces, RMON, and ethernet are the names of the first-level subdirectories of c:\trendsnmp/reports. Double-click on any one of these directory names to make that directory the current directory. The directory name will appear in the Look in edit box.

Designating the Top Level Report Directory

When you first invoke the Report Launcher, the name of the Top Level Report Directory that was in effect when you last closed Report Launcher is used as the initial Top Level Report Directory, and is also made the initial current directory. You can designate a different Top Level Report Directory as follows:

1. In the Report Launcher window, select Report Directories from the View menu:



L dsnmp/reports

Folders

Top Level Report Directory
.. Up one level
Cisco
Interfaces
RMON
ethernet

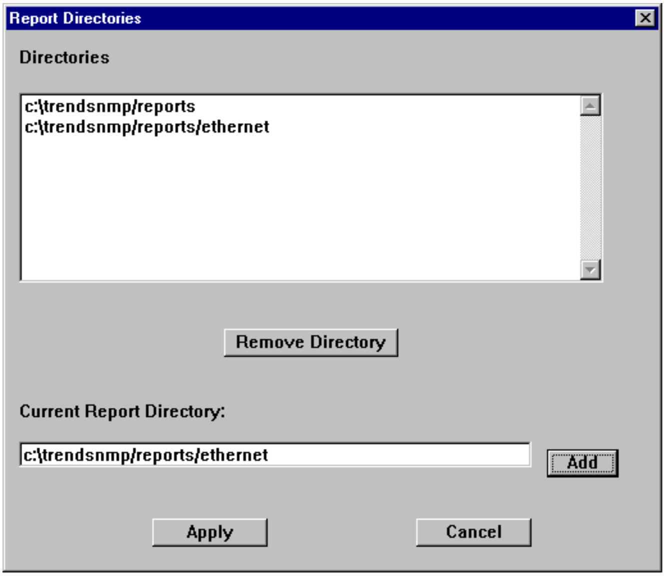
Reports   Time Period:

Daily RMON Ethernet Statistics

Report Types: All Types

Figure 14-5: Report Launcher Window, View Menu

The Report Directories window appears:



c:\trendsnmp/reports
c:\trendsnmp/reports/ethernet

c:\trendsnmp/reports/ethernet

Figure 14-6: Report Directories Window

2. Use this window to add or remove report file directories. The list of directories you can work with appears in the Directories list box.

    1. To add a directory to the list, type the full path name of the directory in the edit box and click Add.

    2. To remove a directory from the list, click on the directory name to highlight it, and then click Remove Directory.

    3. To designate the directory to be used as the Top Level Report Directory (which, initially, is also designated as the current directory), double-click on the directory name in the Directories list box. The directory name will appear in the Current Report Directory edit box. (Alternatively, you can type the full path of the directory name in the Current Report Directory edit box.)

3. Click Apply to apply the changes you have made, close the window, and return to the Report Launcher window. When you return to the Report Launcher window, the directory named in the Current Report Directory edit box of the

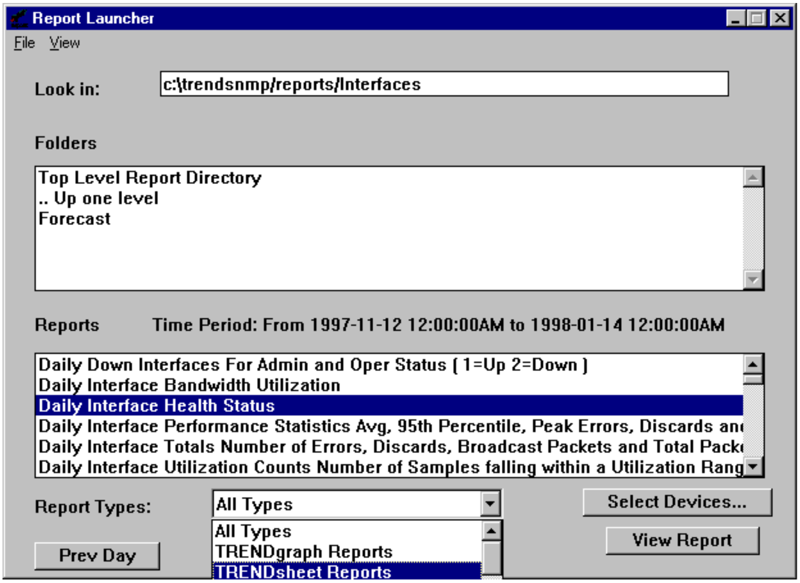
Report Directories window becomes the new Top Level Report Directory and the new current directory (displayed in the Look in box).

Selecting a Report

Hundreds of reports are available in TREND. To make the job of selecting the report to view with Report Launcher easier, you can specify filters to restrict the number of reports, devices, and elements from which you choose.

Follow these steps to tailor the list of reports that appears in the Reports list box in the Report Launcher window:

1. Select the desired report type from the Report Types list box. Figure 14-7 shows sample list box entries:



Look in: c:\trendsnmp/reports/Interfaces

Folders

Top Level Report Directory
.. Up one level
Forecast

Reports Time Period: From 1997-11-12 12:00:00AM to 1998-01-14 12:00:00AM

Daily Down Interfaces For Admin and Oper Status (1=Up 2=Down)
Daily Interface Bandwidth Utilization
Daily Interface Health Status
Daily Interface Performance Statistics Avg, 95th Percentile, Peak Errors, Discards and
Daily Interface Totals Number of Errors, Discards, Broadcast Packets and Total Pack
Daily Interface Utilization Counts Number of Samples falling within a Utilization Rang

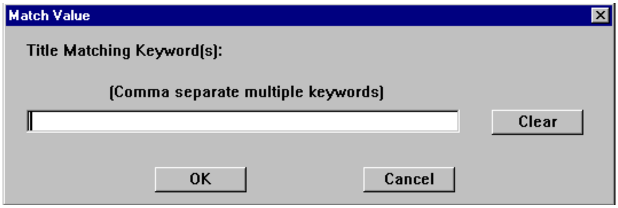
Report Types: All Types

All Types
TRENDgraph Reports
TRENDsheet Reports

Figure 14-7: Report Types List Box

2. You can choose one of the following filters:

Report Type	**Retrieves ...**
No Filter	The default report type, which is all reports that are defined in the current (Look in) report directory).
TRENDsheet	All TRENDsheet (.qss) reports in the current (Look in) report directory.
TRENDgraph	All TRENDgraph (.qgr) reports in the current (Look in) report directory.
Grade of Service	All Grade of Service (.gos) reports in the current (Look in) report directory.
Hourly Reports	All hourly TRENDsheet, TRENDgraph, and Grade of Service reports in the current (Look in) report directory.
Daily Reports	All daily TRENDsheet, TRENDgraph, and Grade of Service reports in the current (Look in) report directory.
Weekly Reports	All weekly TRENDsheet, TRENDgraph, and Grade of Service reports in the current (Look in) report directory.
Monthly Reports	All monthly TRENDsheet, TRENDgraph, and Grade of Service reports in the current (Look in) report directory.

Report Type	Retrieves ...
Title Matching Keywords	The Match Value window appears if you select this filter.  Match Value Title Matching Keyword(s): (Comma separate multiple keywords) Clear OK Cancel All reports in the current (Look in) report directory whose title matches the keywords you specify in the Match Value window are retrieved. Separate multiple keywords by commas. Multiple keywords result in AND-type (rather than OR-type) selection criteria.

3. If the Reports list box has a large number of entries, you can optionally search the list more quickly by selecting the Search Lists option from the View menu (see Figure 14-5). The Search Lists window appears if you select this option:

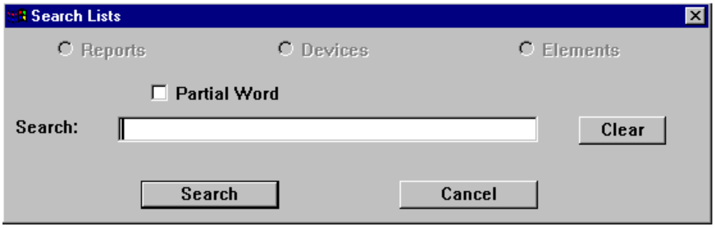


Figure 14-8: Search Lists Window

4. Specify a character string in the Search box.

5. Select the Partial Word check box if the string you enter is not a complete word. (A partial word specification has the effect of sending an SQL WHERE clause with wildcard characters in the search argument. For example, %frame%.)

6. Click Search. The list of entries in the Reports list box in the Report Launcher window is searched, and you are positioned at the first entry that matches the search criteria. (No database access occurs.)

   Continue to click Search to scroll through the entries that match the search string.

   (Click Clear to clear the search string specified in the Search box.)

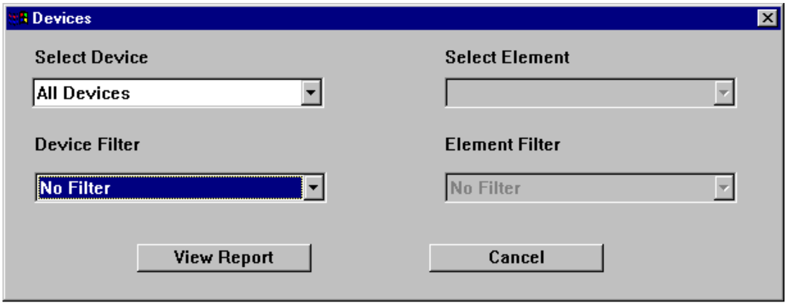
7. Click on the desired report name to select it. (The selected name is highlighted.) Then, click View Report to create the report.

   Alternatively, you can launch the report by double-clicking on its name in the Reports list box.

Selecting Devices and Elements

The default is to include all devices and elements in a report, but you can restrict report contents to selected devices and elements. To do this:

1. Click on the Select Devices button before you launch the report. The Devices window appears:



Select Device: All Devices

Select Element:

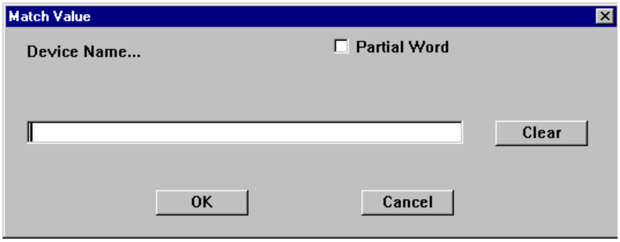
Device Filter: No Filter

Element Filter: No Filter

Figure 14-9: Devices Window

2. Select the device on which you want to report from the entries in the Select Device list box. All Devices is the default.

To make your job of choosing a device easier, you can restrict the entries on the device list by selecting a filter in the Device Filter list box. You can choose one of the following filters:

Filter	**Retrieves ...**
No Filter	No filtering criteria is applied to the list (the default).
Device Name	The Match Value window appears if you select this filter:  Match Value Device Name... □ Partial Word Clear OK Cancel **Figure 14-10: Match Value Window for Device Name** You can specify a device name (such as **`frame25`**) or a partial word string (such as **`frame`**) in the text box. Select the Partial Word check box if you specify a partial word string. The filtering criteria is applied to the Device Name column in the database; the device entries that are retrieved for display in the Select Device list box are restricted to those that match the specified value.
Device Description	The Match Value window for device description appears if you select this filter. This window looks similar to the Match Value window for an element description (see Figure 14-11). The filtering criteria is applied to the Device Description column in the database; the device entries that are retrieved for display are restricted to those that match the specified value.

3. Even after filtering criteria is specified, the list of entries in the Select Device list box can be lengthy. In this case, you can search the list for entries that

match a specified string by selecting the Search Lists option from the View menu (see Figure 14-5). The Search Lists window appears if you select this option (see Figure 14-8).

4. In the Search Lists window, select the Devices radio button, and specify a search string in the Search text box. If the string is a partial word, select the Partial Word check box.

5. Click Search to locate the entries in the Select Device list box that match the specified search value.

   (Click Clear to clear the text in the Search text box.)

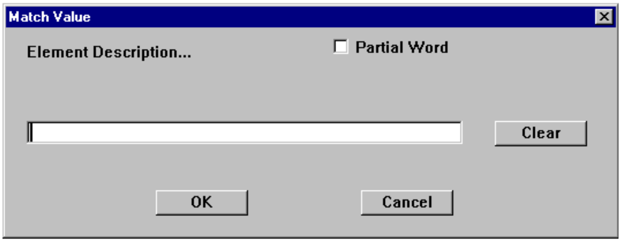
6. Click on the device name to select it. The name of the selected device is highlighted.

   After you select the device, the elements for that device are retrieved from the database and are listed in the Select Element list box.

Note: If you selected All Devices, the Select Elements and Element Filter lists are inactive.

7. If you selected a specific device, you can now select the element on which you want to report from the entries in the Select Elements list box. All Elements is the default. You can restrict the entries on the list by selecting a filter in the Element Filter list box. You can choose one of the following filters:

Filter	**Retrieves ...**
No Filter	No filtering criteria is applied to the list (the default).

Filter	Retrieves ...
Element Name	The Match Value window for the element name appears if you select this filter. The window is similar to the Match Value window for the device name (see Figure 14-10). You can specify an element name (such as **`Async1-1`**) or a partial word string (such as **`Async`**) in the text box. Select the Partial Word check box if you specify a partial word string. The filtering criteria is applied to the Element Name column in the database; the elements that are retrieved for display in the Select Element list box are restricted to those that match the specified value.
Element Description	The Match Value window for element description appears if you select this filter:  Match Value Element Description... □ Partial Word Clear OK Cancel **Figure 14-11: Match Value Window for Element Description** The filtering criteria is applied to the Element Description column in the database; the elements that are retrieved for display are restricted to those that match the specified value.

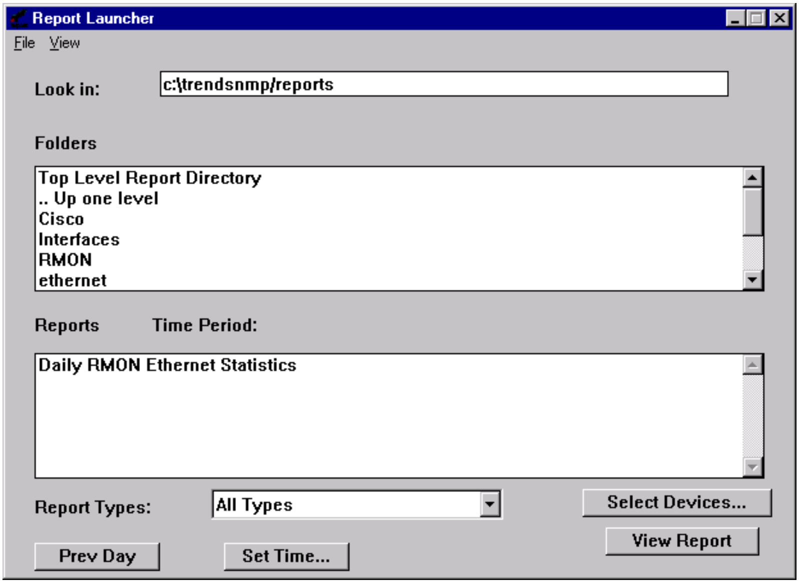
8. Even after filtering criteria is specified, the list of entries in the Select Element list box can be lengthy. In this case, you can search the list for entries that match a specified string by selecting the Search Lists option from the View menu (see Figure 14-5). The Search Lists window appears if you select this option (see Figure 14-8).

9. In the Search Lists window, select the Elements radio button, and specify a search string in the Search text box. If the string is a partial word, select the Partial Word check box.
10. Click Search to locate the entries in the Select Element list box that match the specified search value.

    (Click Clear to clear the text in the Search text box.)

Specifying the Report Time Period

Start and end times are specified for some reports in the report definition file. If so, those times are displayed after the Time Period label in the Report Launcher window.



Look in: c:\trendsnmp/reports

Top Level Report Directory
.. Up one level
Cisco
Interfaces
RMON
ethernet

Daily RMON Ethernet Statistics

Report Types: All Types

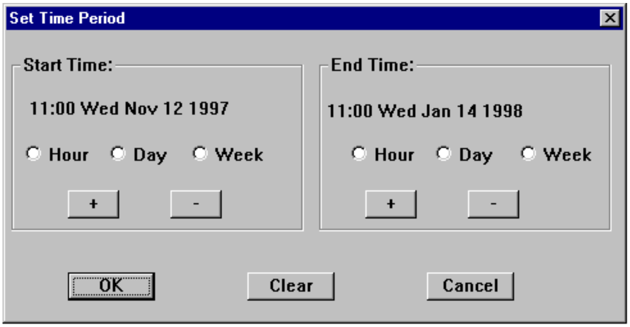
Figure 14-12: Report Launcher Window (Time Period Considerations)

If no start and end times are specified in the report definition file (as in Figure 14-12), you must click Set Time to specify the start and end times.

If a start and end time are displayed, you can also click Set Time to change the times.

You can specify (or change) the hour, day, and week for the Start and End Time. To do this:

1. Click the Set Time button in the Report Launcher window (see Figure 14-12). The Set Time Period window appears:



Start Time: 11:00 Wed Nov 12 1997

End Time: 11:00 Wed Jan 14 1998

Figure 14-13: Set Time Period Window

2. Set the start and end times by clicking the Hour, Day, and Week radio buttons, respectively. Click the + button to increment the selected hour, day, or week. Click the - button to decrement the selected hour, day, or week.

   (Click Clear to clear the displayed Start Time and End Time.)

3. After you set the Start Time and End Time, click OK to close the window. The time period you have specified is displayed after the Time Period label in the Report Launcher window.

If no time period is given for the report in the database or the report definition file, you cannot specify a time period. The Set Time Period window is inactive, and the following text is displayed after the Time Period label in the Report Launcher window:

```
Not Specified in Report
```

You can quickly request a report for the previous period by clicking on the Prev Hour, Prev Day, Prev Week, or Prev Month button (to the left of the Set Time button in the Report Launcher window).

The button is labeled Prev Hour, Prev Day, Prev Week, or Prev Month, respectively, depending on whether an hourly, daily, weekly, or monthly report is selected in the Reports box. In the example (see Figure 14-12), the button is labeled Prev Day because a daily report is selected in the Reports box.

Generating the Report

After you have specified the report, device(s), element(s), and Start and End Times, as appropriate, click the View Report button to generate the report.

Note: The View Report button appears in the Report Launcher window (see Figure 14-12) and in the Devices window (see Figure 14-9). You can launch the report by clicking on the button in either window. (If you want to produce the same report for different devices or elements, you might find it handy to leave the Devices window open and launch the reports from it.)

Refreshing the Report List

If you think the contents of the current report directory has changed (for example, another user has added or deleted report definitions from the report directories you are working with), you can easily refresh the display in the Reports list box.

To do so, select Refresh Reports from the File menu (see Figure 14-3).

15 Report Scheduler

Report Scheduler allows the user to print a report or a series of related reports on a periodic basis. Report Scheduler's automatic printing function processes selected reports to be printed at specified printers or to specified files, at desired times, on daily, weekly, or monthly intervals.

In most networks, there is a set of devices (e.g., core routers) that need to be monitored on a regular basis. Using Report Scheduler, the network staff can receive a series of related reports providing a complete picture of a given network anomaly (e.g., router port utilization threshold exceeded).

Printing a Report

Selecting Report Scheduler from the Main window invokes the main Report Scheduler window:

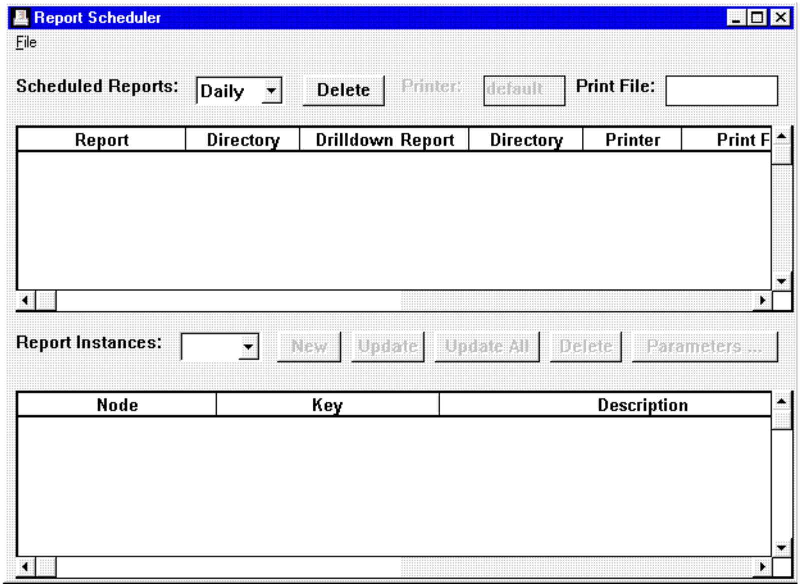
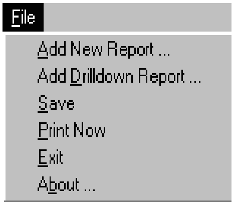


Figure 15-1: Report Scheduler Window

Menu Command Summary

One command menu, **File,** is available from the Report Scheduler window, the options available are:



The **Add New Report...** option under the **File** menu, allows the user to assign an existing report to be printed periodically (see“Adding New Reports” on page 15-4).

The **Add Drilldown Report...** option under the **File** menu, allows the user to link related reports for automatic printing with the main report (see “Report Chaining -- Adding Drilldown Reports” on page 15-4).

The **Print Now** option, under the **File** menu, prints the highlighted report for the previous day's data. The **Save** option should be used before printing.

Note: You must have data (e.g., from the previous day) in order for a report to print.

Scheduled Reports

All report information is presented in the **Table Display** area. If all the information cannot be displayed in a single window, a scroll bar makes it easy to move through the data in the table. The display area is organized into the following areas:

Report	Contains a list of reports selected by the user.
Directory	Contains the directory path where the report is located.
Printer	Contains the name of the postscript printer specified for printing.
File	Contains the name of the file to which the report is to be printed.
Copies	Defines the number of copies of the report to be printed.
ID	The number assigned to the report by Report Scheduler.

Report Instances

All report information is presented here, use the scroll bar to move through the data if it is not displayed in the table. The display area is organized into the following areas:

Node Contains the name of the node associated with that instance of the report. The name default appears if a node is already assigned.

Key Contains the node key number.

Description A description of the node key.

Line # Corresponds to a parameter in the query file which can be manipulated by the user.

Adding New Reports

The **Add New Report** option invokes the File Selection window. See "The Open Command" on page 1-15 for a complete description of the file selection process.

Report Chaining -- Adding Drilldown Reports

Report Scheduler's report chaining capability gives users the ability to generate graphical reports for a series of related network elements, based on a corresponding listing or exception report. There are two general scenarios for use of this feature:

1. Inventory Reports. For example, a report listing all router interfaces is generated each week. Corresponding graphical reports for each router interface display interface health and interface utilization for the same week.

2. A Set of Exception Reports for Devices. For example, a daily report lists Problematic WAN Links on the intranet. Detailed drilldown reports reveal overutilized links, excessive errors, and other components that make the identified links problematic.

Defining Scheduled Reports

Reports selected from the File Selection window are added to the table display. A report is scheduled to print daily, weekly, or monthly depending on whether you select Daily, Weekly, or Monthly in the Scheduled Reports box when you add the new report. You can change the Scheduled Reports field by clicking on the down arrow, and then clicking on the desired print frequency.

To delete a report, click on the report in the table display, click the **Delete** button, and the **Save** option to implement the change.

Defining Report Instances

Additional instances of a report having different parameters can be created and printed using Report Scheduler. If you click the **New** button, another instance will be added. From here you can click the **Parameters** button and change the Report Parameters for that instance (see "Setting Report Parameters" on page 15-6). The original report serves as the template for the new instance.

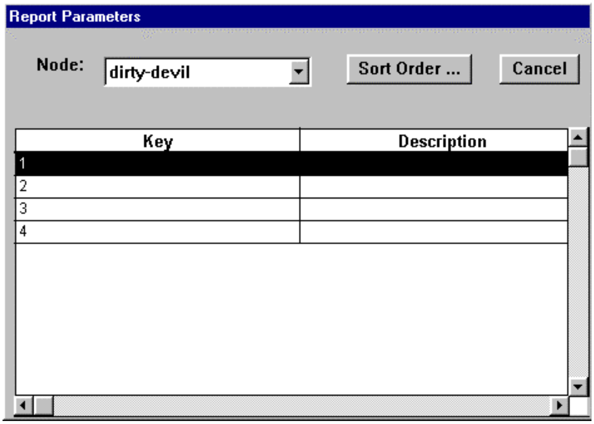
You can also update/change parameters for an instance that already exists. Select the instance you wish to update from the **Report Instances** box by clicking on the down arrow and then selecting the desired instance. Next, click the **Parameters** button, to display the Report Parameters window (see "Setting Report Parameters" on page 15-6). When you have the Report Parameters set the way you desire, you can click the **Update** button in the Print window. This will change the fields in the table display for the instance selected in the **Report Instances** box, and update them with the new values you have chosen.

To delete an instance, simply highlight it in the table display, select the instance you wish to delete from the Report Instances box, and click the **Delete** button.

Note: You must have at least one Report Instance for a report listed in the table display area.

Setting Report Parameters

If you clicked on the **Parameters** button in the Print window, the Report Parameters window is displayed.



Report Parameters

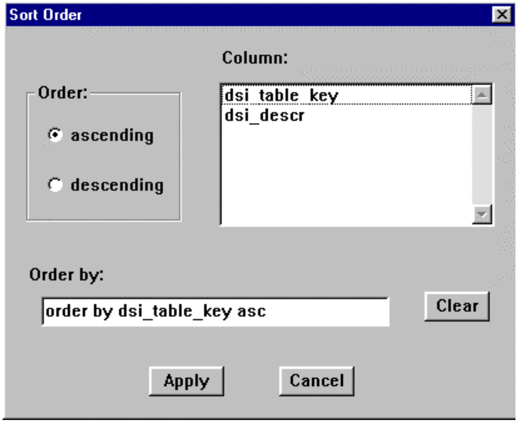
Node: dirty-devil

Key	Description
1	
2	
3	
4	

Figure 15-2: Report Parameters Window

This window allows you to select different nodes, and different keys associated with a particular node. Furthermore, you can change the Sort Order to see reports sorted on a different field, or in ascending rather than descending order (see "TRENDsheet" on page 11-1 for more details on nodes, keys, and sorting).

Display the Sort Order window by clicking on the **Sort Order...** button:



● ascending
○ descending

Order by: order by dsi_table_key asc

Figure 15-3: Sort Order Window

To change the sort order, first delete the existing sort order by clicking on the **Clear** button. Next, select whether the sort key (i.e., table column) should be sorted in ascending or descending order by clicking on the appropriate button in the **Order** box. Select the first sort field from the **Column** box by clicking on the field's name in the box. The key and order you choose will appear on the **Order by** line. You can choose multiple sort keys with a different order for each key. When the sort order is the way you want it, click the **Apply** button to save and apply the changes. To quit this window without saving the changes, click the **Cancel** button.

Output to a File

Reports can also be printed automatically to a postscript file using the **File Print:** option in the **Report Scheduler** window (see Figure 15-1). When the file name is created, extra characters will be attached to it automatically to ensure that each name is unique.

Example

If you want the output to be written to a file named xyz.gif, the actual name of the file will be:

`xyz.`*identifier*`.`*page_number*`.gif`

where:

identifier Is a unique string that is added to the file name to ensure uniqueness.

page_number Is the page of the report shown in this file.

A companion description file is also created that contains information about .gif file contents. The description file is named:

`xyz.`*identifier*`.dsc`

where:

identifier Is the same identifier used in the .gif file name.

TRENDprint File Naming

The extension to the output file name determines the format in which the report file is saved (.ps, .gif, or .bmp).

TRENDprint can accept only one value for the name of the output file or files generated by a scheduled report request. If you define a TRENDprint drilldown sequence, that sequence will generate at least two different reports, and probably more. Of those, the 2nd through nth report will have as many output files as there are device/instance pairs identified in the exception report step of the drilldown sequence.

For example, if a drilldown sequence includes three drilldown reports, and the exception report identifies ten objects to report in more detail, 31 reports are generated. All 31 reports will have the same name. For example, the name `xyz`.*identifier*.*page_number*.`gif` is assigned to 31 files. Moreover, 31 .dsc files are created at the same time.

TREND

16 RMON Support

RMON stands for **R**emote **MON**itoring—a means to observe traffic on a network segment. This is accomplished via an RMON agent, which typically runs on a network device. A device supporting an RMON agent is able to store significant information about the network devices that are talking to each other.

RMON is also the name of the defined data structure (MIB) by which RMON agents order and store the data they collect. Thus, *RMON agents* are those agents that collect and store data defined in the RMON MIB.

TREND components provide data collection, aggregation, and reporting capability for network environments utilizing RMON probes.

RMON data presents added challenges to the collection and storage of network data due to its high volume and dually-indexed table structure. The primary difference between TREND's SNMP and RMON pollers are the data they collect and the methods they employ to collect the data. The RMON polling agent has been designed to collect RMON data and convert it to a TREND format so that it can be processed and used for reporting by generic TREND modules.

TREND includes support for RMON2 network and application layer functionality via AXON Networks Enterprise Communications Analysis Module (ECAM) and

NETscout software Domain View extensions. TREND supports all RMON probes and operates transparently to the user.

RMON Group Support

RMON data polling is set up via the Collect Data application.

TREND's RMON polling agent, `rmon_collect,` supports polling of RMON data. The user can select a polling interval for individual nodes or groups of nodes. rmon_collect supports the *Ethernet* and *Token Ring* RMON MIBs for **History, Host, Matrix and Statistics** groups. TREND assumes that the RMON probes or agents have already been configured using a dedicated RMON real-time application (typically provided by the RMON probe vendor). TREND automatically adjusts to RMON probe configuration changes and no integration with other RMON applications is required.

Group	Description
Statistics	Statistics measured by a probe, including the number and size of packets, broadcasts and collisions. MAC-layer and promiscuous versions of Token Ring RMON Statistics tables also supported.
History	Periodic statistical samples recorded over time, used to analyze trends. MAC-layer and promiscuous versions of Token Ring RMON History tables also supported.
Host	Statistics of the hosts on the network, including MAC addresses of active hosts.
Matrix	A matrix of statistics that tracks conversations between pairs of hosts.

Once TREND creates the host table, it can be sorted on any field, in ascending or descending order, with the desired number of entries selected. Also, all the statistics in the selected hostEntry table rows are available to the user. These abilities exceed the limitations of and therefore preclude the need for the standard topN group collection. In addition, since they are not appropriate for the long term performance profiling done by TREND, alarm, filter, capture and event groups are not polled. Furthermore, because TREND does not distinguish between the Source-Destination and Destination-Source versions of the Matrix table (the table in the database can be sorted and read in any order), it is not necessary to collect both versions of the table from the probe.

Except for `etherStatsEntry`, the supported RMON tables use structures which are unlike the regular SNMP table structure. Most SNMP tables are simple lists of variables, where each instance of a table is one copy of the set of variables (e.g. each instance collects the variables for one interface on a router). Each instance of RMON table is a set of instances grouped by common attributes. For example, a history table instance may consist of a set of history table variable lists. Each of these lists contains the values of variables collected for a specific amount of time (e.g. 50 instances or buckets of the history table variables, each representing 30 seconds of collected data, for a total of 25 minutes.

Note: Do not load an RMON MIB into MIBwalker and define periodic data collection through the MIB map and MIBwalker. If you do, invalid data will be collected.

Tables and Data Tables

The RMON **History, Host,** and **Matrix** groups contain control tables and data tables. The control tables identify individual data tables for each group. Data tables are logically divided into smaller tables according to control table rows. Data tables identify the data sources.

The RMON MIB uses a control table with the history, host and matrix tables to store the parameters that define what each instance of those tables represents. For example, the control table stores the name of the user who created the instance, the interface on the device that is the source of the data, and how many buckets the instance contains.

To correctly collect the data in the history, host and matrix tables, the user actually defines polling on the control table; however, TREND collects both the control table and data table entries each time the control table is collected.

TREND polls control tables, gets data, and augments a key table with information associated with each data source. This information includes the speed and type of the associated RMON probe network interface. TREND polls for only one data table per source. For example, if history tables are setup as separate short and long interval tables, TREND will only poll one instance of the data table, based on a best-fit concept.

If the data source is a *Frontier* Domain, the associated domain description information is also retrieved.

The **Statistics** group does not have a control table, but does contain a data source field. This group’s structure is dealt with in the same manner as the History, Host, and Matrix tables.

Polling is performed by node, view, or type, by individual node or by groups of nodes. Four tables are created from information from each group’s control table and corresponding key table. This includes History, Host, Matrix and Statistics. In addition, a data table is created with values from all probe data tables polled. A Matrix group control table, for example, will be named rmon_matrix and the data table will be `rmon_matrix_data.`

Database tables for the supported RMON tables are pre-loaded into the database as part of the installation process.

Polling Frequency

The polling frequencies you set for the various tables should take into consideration the size of the table to be returned and the impact this will have on your probe, your network and your database. You should also consider how often you are going to look at data and what reporting interval makes sense. For example, if you are not going to look at data in intervals of less than an hour there is no real reason to collect it more frequently. Also, see the recommended polling intervals in the descriptions of the reports included with TREND.

Note: It is highly recommended that Host and Matrix data be collected on a selective basis with a polling interval of one hour. The vast quantity of data returned due to the size of these tables has an impact on network utilization and database performance. The amount of database space used will also increase dramatically when collecting Host and Matrix table data.

History Group Polling

The setting of polling intervals for the History group requires special consideration. The RMON history structure provides a major benefit to managers who want to report on relatively fine-grain network activity data. Normally, obtaining one minute data samples would require polling once every minute. In a large network where many devices are to be polled, such frequent polling not only utilizes measurable management station and network bandwidth, but the skew between samples (i.e., imprecision of the one minute interval will be large. Fortunately, RMON History group data consists of a set of samples maintained by the RMON agent. Hence polling on a leisurely basis can still yield fine grain data.

Historys differ in the number of samples (buckets) and intervals (bucket width). If we assume a short History (fifty 30-second buckets) and poll every 20 minutes, the result is 50 data samples, each corresponding to a different 30 second period, and each containing delta (TREND rate table) values for each statistic. Raw data for the History group is not inserted into the database, rather it is already rolled up in the form of rate data. Data is therefore directly inserted into the rate table with timestamps set.

The polling interval used for History data should be based on the total timespan covered by the samples maintained by the RMON agent or probe. The typical default short History spans 25 minutes (i.e., 50 samples or buckets of 30 seconds each). If these default settings are used, polling on a 20 minute basis should ensure collection of 30-second grain data without gaps while polling once a day should ensure collection of one hour grain data without gaps.

Note: History group polling should be defined with an interval that is less than the total timespan covered by the probe. In this way, variances in network and probe response times are transparent due to collection of overlapping data. The duplicate data is not inserted into the TREND database.

Probes and agents can support multiple Historys for a given data source. It is usually recommended that there be at least two History entries per monitored interface. An entry with a short sample period of 30 seconds enables the detection of sudden changes in traffic patterns. A long sample of 30 minutes, on the other hand, reveals the steady-state behavior of the interface.

Since TREND will collect the short history data and aggregate it over time, probe resources can be conserved by only configuring a single History where the interval yields the finest grain data of interest. TREND's capability to provide min, max and avg values is especially significant as applied to reporting on RMON History groups. While min and max reveal short-term changes over hours, avg provides a view of long-term variations over days.

Note: TREND will only poll for a single history instance for each data source (i.e., probe network interface or probe network interface coupled with Frontier Domain). If multiple Histories are configured, TREND will select one on a best-fit basis. The best fit can be found by multiplying the number of buckets by the bucket width to calculate the total span for History. The span selected is the smallest one greater than the polling interval. The RMON polling process, rmon_collect, will analyze each history table instance and put information in the file $DPIPE_HOME/

tmp/trend.log informing you if the polling intervals defined for your history tables are appropriate.

For example, if an instance of the history table has 70 buckets representing one minute each, that instance spans across an hour period (actually 70 minutes). There is no reason to poll it more frequently. **DeskTalk recommends this polling scenario**. Conversely, if the history table has an instance with 50 buckets of 30 seconds each, only 25 minutes are represented and polling once an hour will lose data collected in the other 35 minutes.

Even if only polled once per day, rate tables will contain data samples timestamped every 20 minutes. If polling once per hour, select long History. Each hour, 50 buckets will be collected of which 48 will be thrown out, leaving 2 samples to be stored in the database. If polling once each 20 minutes, select short History, adding approximately 40 samples to the database (note that the actual number may not be exactly 40 since the polling interval is not precise in a network environment).

Aggregating Host and Matrix Tables and Managing Size

The Host and Matrix tables have the potential to be extremely large and unwieldy. To minimize this possibility the TREND rollup process, TRENDit suppresses zero rate rows in the raw tables (pertain to pairs of devices that did not exchange any data during the polling period). TRENDit can also be configured to delete raw data below a specified threshold as it is rolled up. See "Man Pages" on page A-1 for details on using TRENDit.

Note: If noddy talked to notable at some point, an entry exists in the RMON probe's matrix table for noddy-notable. However, they may never or rarely communicate again, so the probe is returning a zero value row for noddy-notable each time the matrix table is collected. By discarding

such rows from the database before they are aggregated and carried into the rate, hourly, daily and weekly tables, the already large matrix table kept to the minimum number of meaningful rows.

TREND RMON Reports

All TREND reports have been developed using TREND reporting tools, with no special code having been written. Some of these reports, however, required the construction of a database view to extend the database schema.

Viewing RMON Tables in Data Manager

When you first look at the RMON tables in Data Manager, there appear to be pairs of raw data tables. However, a closer look will reveal some differences:

- The tables whose Object names end in data and whose SQL names end in data_ are the actual data tables. The other similarly named table is the control table. For example, the table with Object name rmon_history and SQL name rmon_hist_ is the control table for the history table. The table with Object name rmon_history-data and SQL name rmon_histdata_ is the data table for the history table.
- The Rollup column value for the control tables is set to no. This must not be modified.
- The ethernet statistics table and token ring statistics tables share a common control table, rmon_ethernet_statistics.
- The ethernet history table and token ring history tables share a common control table, rmon_history.

Domains

The RMON probes made by Netscout software implement domains that allow the user to create subsets of the RMON tables which only contain data that pertains to the defined domains. Domains can filter on protocols and applications, so a user can create a domain that only collects statistics about IP traffic for example, and by applying that IP domain to the history table, see history statistics generated exclusively by IP traffic on the monitored network.

TREND understands Netscout RMON probe domains, and allow the user to create reports that take advantage of them. The key tables for the RMON data tables include a filter field. By creating a report which specifies the name of the filter, a report can isolate and display domain-specific information.

For probes which do not support domains, TREND uses the value ALL in the filter field. This value implies that the data is not limited to a specific domain.

Note: In Collect Data, the Tools button has a TRENDgraph and TRENDsheet option. Each of these allow you to do a quick, unformatted dump of the data in the selected mib group’s raw, rate, hourly, daily or weekly table. For RMON, the table you would see by doing this is the control table rather than the data table and consequently, by default, only raw data will be available. (See “About Data and Data Tables in TREND” on page 10-2.**)**

TREND

17 RMON2 Support

The RMON2 MIB is an extension of the RMON MIB. It includes all the data objects defined in the RMON MIB, as well as many others. *RMON2 agents* are those agents that can store some or all of the information that is unique to the RMON2 MIB.

The RMON2 DataPipe (poller) vastly increases TREND’s RMON reporting capability, both for RMON and for RMON2.

Note: In the sections below, each category of table shows two distinct tables. Each of these pairings represents a data table (in which collected data is actually stored) with its associated control table (which defines the format of the data table). Data tables are identified by the suffix data in the Database Name column. Control tables have no suffix.

Source Data Available from RMON2

RMON Data

The RMON2 DataPipe supports extensions to standard TREND RMON tables. The only difference between the regular RMON tables (identified by the prefix rmon_) and the extended RMON tables (identified with the RMON2 prefix rmone_) is that the extended tables contain Dropped Frames and Table Creation Time information. To avoid duplicate collection, be sure to collect *rmone_* tables only from probes that support the RMON2 extensions.

Note: You should use only the RMON2 DataPipe to collect data from agents that support the RMON2 MIB. The table extensions described below cannot be collected from regular RMON agents. Use TREND's standard RMON DataPipe to poll agents that support the standard RMON MIB.

Supported RMON Table Extensions

There are twenty RMON table extensions supported by the RMON2 DataPipe:

Category	Database Name	Alias Name	MIB Table
Ethernet Statistics	rmone_ethstats_	rmone_ethernet-statistics	Control information from EtherStatsEntry
Ethernet Statistics	rmone_ethstatsdata_	rmone_ethernet-statistics-data	Statistics from EtherStatsEntry and EtherStats2Entry

(1 of 3)

Category	Database Name	Alias Name	MIB Table
Token Ring Promiscuous statistics	rmone_trpstas_	rmone_token-ring- promiscous -statistics	Control information from tokenRingPStatsEntry
Token Ring Promiscuous statistics	rmone_trpstasdata_	rmone_token-ring-promiscous-statistics-data	Statistics from tokenRingPStatsEntry and TokenRingPStats2Entry
Token Ring MAC statistics	rmone_trmstas_	rmone_token-ring-maclayer-statistics	Control information from tokenRingMLStatsEntry
Token Ring MAC statistics	rmone_trmstasdata_	rmone_token-ring-maclayer-statistics-data	Statistics from tokenRingMLStatsEntry and TokenRingMLStats2Entry
Ethernet History	rmone_hist_	rmone_history	HistoryControlEntry and historyControl2Entry
Ethernet History	rmone_histdata_	rmone_ethernet-history-data	EtherHistoryEntry
Token Ring MAC History	rmone_hist_	rmone_history	HistoryControlEntry and historyControl2Entry
Token Ring MAC History	rmone_histtrmdata_	rmone_tokenring-maclayer-history-data	TokenRingMLHistoryEntry
Token Ring Promiscuous History	rmone_hist_	rmone_history	HistoryControlEntry and historyControl2Entry
Token Ring Promiscuous History	rmone_histtrpdata_	rmone_tokenring- promiscuous -history-data	TokenRingPHistoryEntry
Host	rmone_host_	rmone_host	HostControlEntry and hostControl2Entry
Host	rmone_hostdata_	rmone_host-data	HostEntry
Matrix	rmone_matrix_	rmone_ matrix	MatrixControlEtnry and matrixControl2Entry
Matrix	rmone_matrixdata_	rmone_ matrix -data	MatrixSDEntry
Token Ring Stations	rmone_trstation_	rmone_token-ring-station-entry	RingStationControlEntry and ringStationControl2Entry

(2 of 3)

Category	Database Name	Alias Name	MIB Table
Token Ring Stations	rmone_trstationdata_	rmone_token-ring-station-entry -data	RingStationEntry
Token Ring Source Routing	rmone_trroutestats_	rmone_token-ring-routing-stats-entry	Control information from SourceRoutingStatsEntry
Token Ring Source Routing	rmone_trroutestatsdata_	rmone_token-ring-routing-stats-entry -data	Statistics from sourceRoutingStatsEntry and sourceRoutingStats2Entry

(3 of 3)

Note that in the above table, information from some of the table extensions is stored in control tables. If, for example, you want to include Dropped Frames as part of regular RMON data report, you need to create a view between the data table and the control table.

RMON2 Data

TREND defines RMON2 data as data tables that exist in the RMON2 MIB, but do not exist in the RMON MIB.

Supported RMON2 Tables

There are ten defined RMON2 tables supported by the RMON2 DataPipe:

Category	Database Name	Alias Name	Mib Table
Protocol statistics	rmon2_stats_	rmon2_stats	protocolDistControlEntry
Protocol statistics	rmon2_statsdata_	rmon2_stats-data	protocolDistStatsEntry
Network Layer Host	rmon2_nlhost_	rmon2_nlhost	hlHostControlEntry

Category	Database Name	Alias Name	Mib Table
Network Layer Host	rmon2_nlhostdata_	rmon2_nlhost-data	nlHostEntry
Application Layer Host	rmon2_alhost_	rmon2_alhost	HlHostControlEntry
Application Layer Host	rmon2_alhostdata_	rmon2_alhost-data	alHostEntry
Network Layer Matrix	rmon2_nlmatrix_	rmon2_nlmatrix	hlMatrixControlEntry
Network Layer Matrix	rmon2_nlmatrixdata_	rmon2_nlmatrix-data	nlMatrixSDEntry
Application Layer Matrix	rmon2_almatrix_	rmon2_almatrix	hlMatrixControlEntry
Application Layer Matrix	rmon2_almatrixdata_	rmon2_almatrix-data	AlMatrixSDEntry

DataPipe Components

The RMON2 DataPipe uses SNMP as its transport protocol, and the data it collects does not require translation to be inserted into the TREND database. The data tables required for RMON2 are installed as part of the DataPipe, allowing RMON2 data to be inserted directly into the database.

There are, however, two unique components of the RMON2 DataPipe:

- The RMON2 collector—**dpipe_rmon2**—is the tool that polls an RMON2-manageable device and collects the RMON2 data the device has stored.
- The Protocol Definition Interface is a graphical user interface that allows TREND users to populate the dsi_protocols table.

These components are described in the sections below.

The dpipe_rmon2 Polling Agent

How TREND polls a device (node) is determined by how that device is defined in the dsi_node_list table. If a device is defined as being an RMON capable device, TREND launches RMON_collect, which is a *parent collector*. The parent collector does not perform the actual polling of the device. Instead, the parent collector determines precisely what kind of device the target is, and launches the appropriate *child collector*, which, in turn, polls the target device. In the case of RMON2, the child collector is named **dpipe_rmon2.**

The dpipe_rmon2 collector collects the data stored by the RMON2 agent, and stores that data in the TREND database.

The Protocol Definition Interface

When dpipe_rmon2 collects RMON2 data, the data is organized by protocol. In this context, a protocol defines an exact description of how two devices communicate with each other. RMON2 makes it possible to identify not only which devices on a network spoke to each other, but also how the conversation took place.

However, the collected data only identifies these protocols by their OID strings. Because these OID strings are not readily understandable, TREND allows you to assign them meaningful names. Then, when reports are generated from the collected data, the conversation protocols are recognizable.

TREND assigns names to the OID strings via a translation table, **dsi_protocols**. The dsi_protocols table associates the OID strings with their appropriate name identifier.

Note: RMON2 protocols have a defined naming convention. See “Combined Protocol Identifier and Parameter String Values” on page 17-16**.**

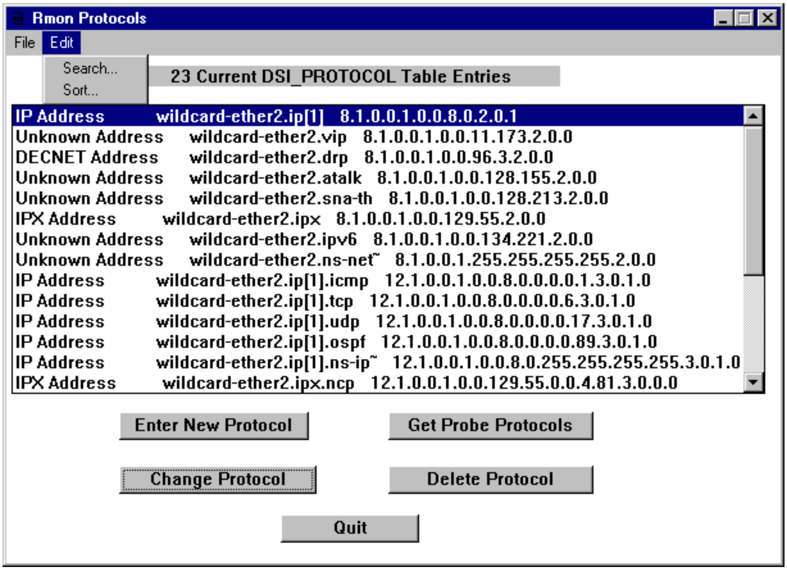
In most cases, it is not necessary for TREND users to know the RMON2 naming convention, or to manually populate the dsi_protocols table. The RMON2 DataPipe includes a graphical interface that automates the population process.

The Protocol Definition Interface is launched from the command line. At the command line:

1. Change directories to **TRENDsnmp\bin**.
2. Type the following:

   `r2proto`

This brings up the RMON Protocols window:



23 Current DSI_PROTOCOL Table Entries

```
IP Address        wildcard-ether2.ip[1]   8.1.0.0.1.0.0.8.0.2.0.1
Unknown Address   wildcard-ether2.vip   8.1.0.0.1.0.0.11.173.2.0.0
DECNET Address    wildcard-ether2.drp   8.1.0.0.1.0.0.96.3.2.0.0
Unknown Address   wildcard-ether2.atalk   8.1.0.0.1.0.0.128.155.2.0.0
Unknown Address   wildcard-ether2.sna-th   8.1.0.0.1.0.0.128.213.2.0.0
IPX Address       wildcard-ether2.ipx   8.1.0.0.1.0.0.129.55.2.0.0
Unknown Address   wildcard-ether2.ipv6   8.1.0.0.1.0.0.134.221.2.0.0
Unknown Address   wildcard-ether2.ns-net˜   8.1.0.0.1.255.255.255.255.2.0.0
IP Address        wildcard-ether2.ip[1].icmp   12.1.0.0.1.0.0.8.0.0.0.0.1.3.0.1.0
IP Address        wildcard-ether2.ip[1].tcp   12.1.0.0.1.0.0.8.0.0.0.0.6.3.0.1.0
IP Address        wildcard-ether2.ip[1].udp   12.1.0.0.1.0.0.8.0.0.0.0.17.3.0.1.0
IP Address        wildcard-ether2.ip[1].ospf   12.1.0.0.1.0.0.8.0.0.0.0.89.3.0.1.0
IP Address        wildcard-ether2.ip[1].ns-ip˜   12.1.0.0.1.0.0.8.0.255.255.255.255.3.0.1.0
IPX Address       wildcard-ether2.ipx.ncp   12.1.0.0.1.0.0.129.55.0.0.4.81.3.0.0.0
```

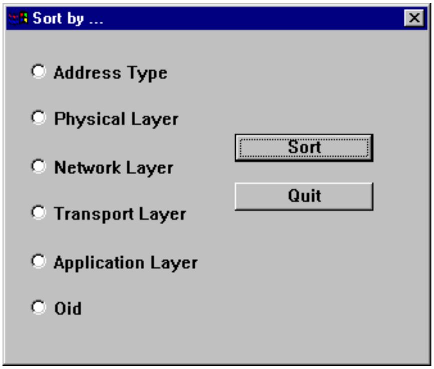
Figure 17-1: RMON Protocols Window

The interface window displays all of the RMON2 protocols defined in dsi_protocols. The window appears empty the first time you launch the interface.

You can search and sort this display.

To sort:

1. Select Sort from the Edit menu. The Sort By window appears:



Sort by ...

○ Address Type

○ Physical Layer

○ Network Layer

○ Transport Layer

○ Application Layer

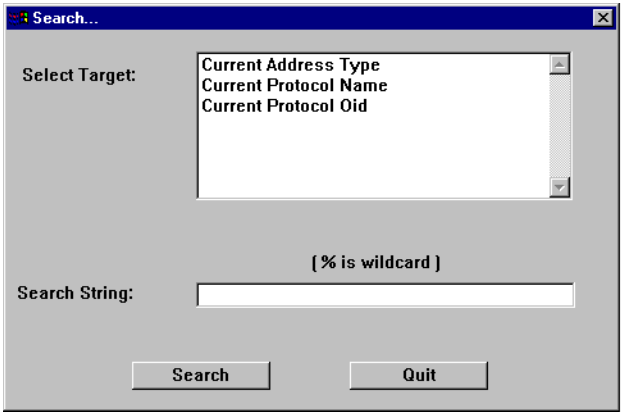
○ Oid

Figure 17-2: Sort By Window

2. Select a sort variable and click Sort.

To search:

1. Select Search from the Edit menu. The Search window appears:



Search...

Select Target:

Current Address Type
Current Protocol Name
Current Protocol Oid

(% is wildcard)

Search String:

Figure 17-3: Search Window

2. Select a target, enter a search string, and click Search.

Using the Protocol Definition Interface to Populate dsi_protocols

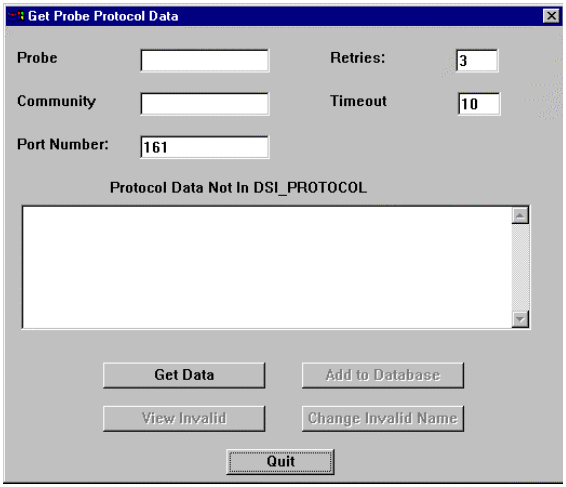
Once the Protocol Definition Interface is running, you can use it to populate dsi_protocols. However, before you start the definition process, it is a good idea to make a note of the different kinds of RMON2 manageable devices you have on your network. There are two reasons for this:

- Though there are approximately 5,000 defined RMON2 protocols, you may not need to use more than a few of them, depending on the reports you wish to generate. Since the size of the dsi_protocols table can affect TREND performance, you probably will not want to enter every single protocol available on your network. In fact, it is recommended that dsi_protocols contain only those protocols that appear in your reports.
- Though some networks have hundreds, or even thousands, of RMON manageable devices, most of these devices do not support RMON2. In addition, even if your network has many RMON2 capable devices, it is likely that you have only a few kinds (models) of devices with RMON2 agents.

Since the device manufacturer defines which protocols are collected by the RMON2 agent, you only need to define (get) the protocols once for each device model.

For example, a really large network might include hundreds of Netscout probes, all of the same model. Since the data set collected by each of these probes is identical to the others, getting the protocols from one probe is the same as getting the protocols from all probes.

When you are ready to get the protocols from an RMON2 manageable device, launch the Protocol Definition Interface, and click the Get Probe Protocols button. The Get Probe Protocol Data window appears.



Probe

Community

Port Number: 161

Retries: 3

Timeout 10

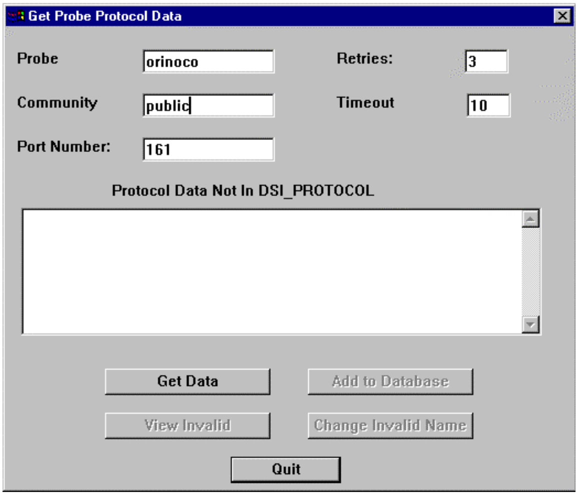
Figure 17-4: Get Probe Protocol Data Window

In order to get probe protocols, you need to supply the IP name (or address) and community string of an RMON2 capable device. Enter the name of the device in the Probe field, and the community string in the Community field.

Optionally, you can also:

- Specify a port number in the Port Number field.
- Define the number of retries in the Retries field.
- Define how much time must pass before the get request is timed out in the Timeout field.

Figure 17-5 shows the window with a node specified.



Get Probe Protocol Data

Probe: orinoco

Community: public

Port Number: 161

Retries: 3

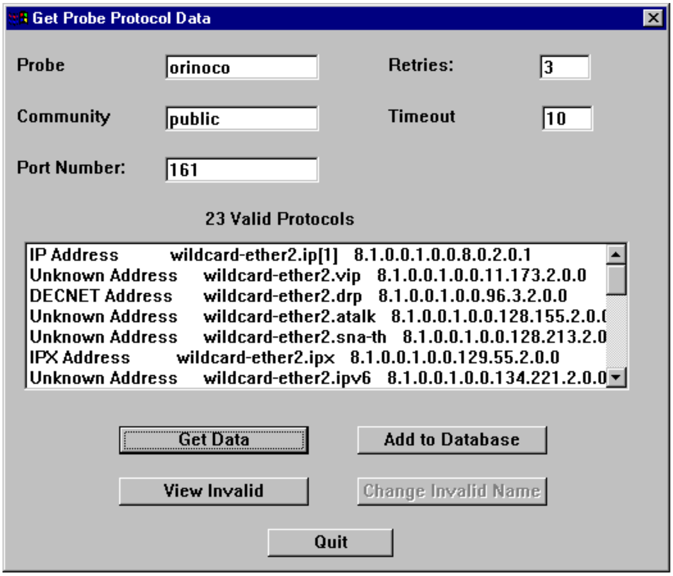
Timeout: 10

Protocol Data Not In DSI_PROTOCOL

Figure 17-5: Get Probe Protocol Data Window (with node identified)

When you have specified the node, click Get Data. The Protocol Definition Interface discovers all of the RMON2 protocols being collected by the target device (actually, by its RMON2 agent).

The protocols also appear in the Get Probe Protocols Data window.



Get Probe Protocol Data

Probe: orinoco

Community: public

Port Number: 161

Retries: 3

Timeout: 10

23 Valid Protocols

```
IP Address        wildcard-ether2.ip[1]   8.1.0.0.1.0.0.8.0.2.0.1
Unknown Address   wildcard-ether2.vip   8.1.0.0.1.0.0.11.173.2.0.0
DECNET Address    wildcard-ether2.drp   8.1.0.0.1.0.0.96.3.2.0.0
Unknown Address   wildcard-ether2.atalk   8.1.0.0.1.0.0.128.155.2.0.
Unknown Address   wildcard-ether2.sna-th   8.1.0.0.1.0.0.128.213.2.0
IPX Address       wildcard-ether2.ipx   8.1.0.0.1.0.0.129.55.2.0.0
Unknown Address   wildcard-ether2.ipv6   8.1.0.0.1.0.0.134.221.2.0.0
```

Figure 17-6: Get Probe Protocols Data Window (After a Get)

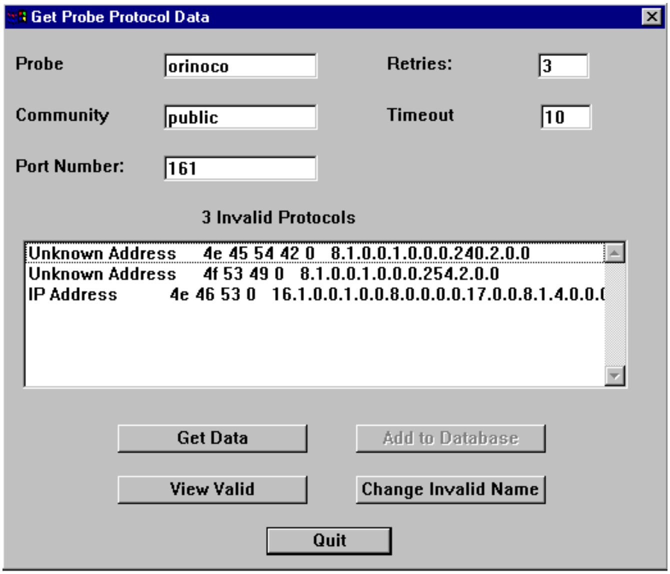
In Figure 17-6, we see that the Protocol Definition Interface was able to identify 23 valid protocols. This means that the target device responded to the get request with 23 different OIDs, each associated with the correct protocol name (as defined by the RMON2 protocol naming convention), and that none of these 23 protocols already existed in dsi_protocols. The leftmost column describes each OID's address type. If the address type is not know, **Unknown** is displayed. Since these 23 protocols are valid, they should be inserted into dsi_protocols. To do this, click Add to Database.

In Figure 17-6, note that the View Invalid button is enabled. A get request can also return invalid protocols. Why this happens, and how to deal with it is described below.

Identifying and Renaming Invalid Protocols

When you use the Protocol Definition Interface to send a get request, it is possible, even likely, that the request will return at least some invalid protocols. When this happens, the View Invalid button is enabled. You can click this button to see any protocols that have not been recognized and accepted by the Protocol Definition Interface.

This is shown in Figure 17-7:



Probe: orinoco
Retries: 3
Community: public
Timeout: 10
Port Number: 161

3 Invalid Protocols

```
Unknown Address     4e 45 54 42 0   8.1.0.0.1.0.0.0.240.2.0.0
Unknown Address     4f 53 49 0   8.1.0.0.1.0.0.0.254.2.0.0
IP Address        4e 46 53 0   16.1.0.0.1.0.0.8.0.0.0.0.17.0.0.8.1.4.0.0.
```

Figure 17-7: Invalid Protocols Display

The Protocol Definition Interface defines a protocol as being invalid for the following reasons:

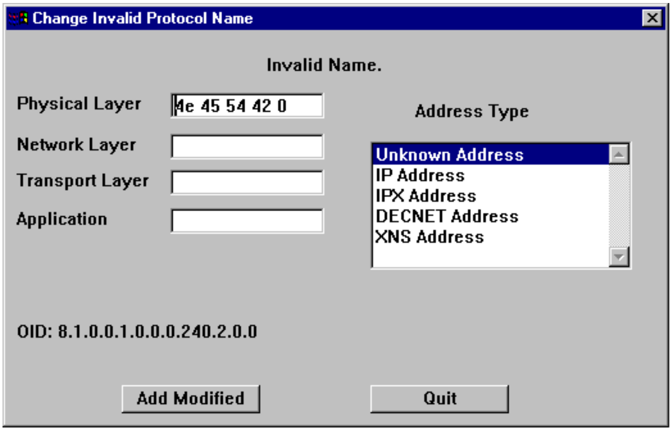
- The protocol already exists in dsi_protocols.
- The device manufacturer has assigned a nonsensical name to the protocol OID.
- The device manufacturer has not assigned any name to the protocol OID.

- The protocol OID cannot be properly identified

As explained in "Using the Protocol Definition Interface to Populate dsi_protocols" on page 17-9, it is unnecessary, even undesirable to include all available protocols in dsi_protocols. In most cases, you can safely ignore any invalid protocols returned by the interface. However, if a protocol you wish to report on is returned as invalid, you can rename it in such a way that the Protocol Definition Interface recognizes it as being valid.

The procedure for this is described below.

1. Click View Invalid to display the invalid returns.
2. Click one of the displayed protocols to highlight, then click Change Invalid Name. The Change Invalid Protocol Name window appears.



Physical Layer 4e 45 54 42 0

Unknown Address

OID: 8.1.0.0.1.0.0.0.240.2.0.0

Figure 17-8: Change Invalid Protocol Name Window

3. To rename the protocol, you can either:
   - Input the correct name, using the RMON2 naming convention.
   - Enter any meaningful name you choose (to make it recognizable to you).

   However, if you choose to enter an unconventional name, the following rules apply:

- You must enter an alphanumeric value of up to eight characters in each of the four fields.
- Only the Application (layer) field can include a period.

4. Once you have defined the name and address type that you wish to apply, click Add Modified. The Protocol Definition Interface adds the newly defined protocol to dsi_protocols.

Note: For more information on the RMON2 naming convention, see "Combined Protocol Identifier and Parameter String Values" on page 17-16**.**

Deleting Protocols from the dsi_protocols Table

If you regularly get protocols, the dsi_protocols table can quickly become unwieldy, and can have an adverse effect on polling efficiency. It is a good idea to periodically clean it up. Make a list of those protocols that appear in your reports, and use the Protocol Definition Interface to delete any unnecessary protocols.

1. Launch the Protocol Definition Interface, as described in "The Protocol Definition Interface" on page 17-6.
2. Highlight the protocol you wish to delete, and click Delete Protocol. The protocol is deleted.

Note: The Protocol Definition Interface only allows you to highlight one protocol at a time. For this reason, you can only delete protocols one by one.

Combined Protocol Identifier and Parameter String Values

In general, protocols used in conversations about RMON2 are only identified by the dotted name notation. However, the dotted name notation is ambiguous when identifying entries in table `protocolDirTable`, since entries are identified by both the value of a protocol identifier string and a protocol parameter string. Also, it is possible for a protocol to have multiple encapsulations under a parent protocol. To solve these problems, there is an extended form of the dotted name format, which can be used to identify entries in the protocol directory table.

The format of extended dotted name notation is shown in Figure 17-9.

```
<extendedDottedName> = ["wildcard-"]<qualifiedProtoName>["."<qualifiedProtoName>]..

<qualifiedProtoName> =  name[<ecapVal>][<parmVal>]

<ecapVal>= "("uint")"

<parmVal>= "["uint"]"

Where:
   name - protocol name from PI definition
   uint - unsigned integer between 0 and 4G-1
```

Figure 17-9: Extended Dotted Name Notation

The wildcard- prefix is specified when the wildcard function has been specified with a base protocol layer identifier. An *ecapVal* qualifier is only specified if the protocol has more than one encapsulation under a parent. A *parmVal* qualifier is only specified if the value of the protocol parameter is nonzero. For example, the IP protocol has parameter *countFragments*. This parameter is bit zero, and thus has a value of one when turned on. Putting it all together, to specify that the wildcard function has been selected and the *countFragments* parameter has been selected for the UDP over IP over Ethernet-II encapsulation, the following extended dotted name is used:

wildcard-ether2.ip[1].udp

This specifies a value of ‘01000001 00000800 00000011’h (1.0.0.1.0.0.8.0.0.0.0.17) for the protocol identifier string and a value of ‘000100’h (0.1.0) for the protocol parameter string.

18 TRENDcopy

TRENDcopy allows the trendadm to copy data from one database to another.

Note: Data transported by TRENDcopy goes from data table to data table. Data copied in this way is only accessible by TREND. The TRENDexporter (see “TRENDexporter” on page 20-1) **allows you to export data in ASCII text format.**

TRENDcopy can:

- Copy all the data in a database table and add that data to a similarly named table in another TREND database.
- Copy only specified columns in a data table and insert them in another database.
- Create a new data table in a database using information copied from a table in another database.
- Filter the data it copies by type, view, date, or time of day.
- Filter the data it copies by columns in a data table or key table.
- Copy only aggregated or summarized data.

TRENDcopy has two common applications:

- In a TREND system that uses Satellite Servers to clean raw data, use TRENDcopy to copy the cleaned data from a TREND Satellite Server to the TREND Primary Server.
- In a TREND system that includes an Archive Server, use TRENDcopy to copy data from the Primary Server to the Archive Server.

Note: A Sybase interfaces file defining the target server must exist on the server that issues the TRENDcopy command. See "Creating the Sybase Interfaces File on UNIX Systems" on page D-8 **or** "Creating the Sybase Interfaces File on Windows Systems" on page D-15 **for details.**

Be aware that TRENDcopy has the following conditions:

- TRENDcopy only copies the definitions of key and data tables to another database when the key table of the destination table is missing an identity column (such as a Report Package demo). In this case, use either the BCP or BCPfile utility to complete the transfer.
- TRENDcopy only copies a source data table with a corresponding key table that is a view when the destination table has a key table present and is also constructed as a view.
- TRENDcopy copies a source data table constructed as a view to an existing destination data table constructed as a view; otherwise, it creates the destination data table as a table.

TRENDcopy Syntax

```
trendcopy   [ -b  length  unit ]
            [ -c  column_name:exp:value ]
            [ -C  key_column_name:exp:value ]
            [ -d  debug_level  ]
            [ -e  yyyymmdd[:yyyymmdd] ]
            [ -f   day_of_week[:day_of_week] ]
            [ -g  hour[:hour] ]
            [ -l  length  unit ]
            [ -m  minimum_threshold_for_bcp_transaction(100) ]
            [ -M  maximum_threshold_for_bcp_transaction(100) ]
            [ -n  node_type ]
            [ -p  polling_list_id ]
            [ -r  table_type ]
            [ -R table_type ]
            [ -s  source_server]
            -S  target_server
            [ -t  source_table:destination_key_table:destination_table ]
            [ -T  source_topology_database ]
            [ -u ]
            [ -U  user_name ]
            [ -v  view_name ]
            [ -V ]
            [ -w  high_water_mark(90) ]
```

TRENDcopy is executed via a command line interface. The following options can be used:

-b Copies data for the length of the time period specified with the **-l** option up to the baseline time specified with the **-b** option. *length* can be any integer, and *unit* can have one of the following values:

h	hour
d	day
m	month
y	year

See "Using the -b and -l Options To Copy Data" on page 18-13.

-c Provides filtering by data table columns. See "Using the -c and -C Options To Filter Data by Column" on page 18-10.

-C Provides filtering by key table columns. See "Using the -c and -C Options To Filter Data by Column" on page 18-10.

-d Sets the debug output level. Valid values are 1, 2, and 3. The higher the number, the more detailed the information. Debug output is written to the standard output destination. This option is for development purposes.

-e Provides filtering by date. A date is specified in the format *yyyymmdd*.

See "Using the -e, -f, and -g Options To Filter Data by Date" on page 18-11.

-f Provides filtering by day of week. Valid weekday values are:

su	Sunday
mo	Monday
tu	Tuesday
we	Wednesday
th	Thursday
fr	Friday
sa	Saturday

See "Using the -e, -f, and -g Options To Filter Data by Date" on page 18-11.

-g Provides filtering by hour of the day. Valid hour values are 1-24, where 24 means 12 midnight.

See "Using the -e, -f, and -g Options To Filter Data by Date" on page 18-11.

-l Length of the time period to copy. This option can only be used with the **-b** option. *length* can be any integer, and *unit* can have one of the following values:

h	hour
d	day
m	month
y	year

See "Using the -b and -l Options To Copy Data" on page 18-13.

-m Minimum threshold for a bulk copy packet (bcp) transaction. The default is 100 rows.

-M Maximum threshold for a bulk copy packet transaction. The default is 100 rows.

-n Provides filtering by node type. If this option is used, the **-U** option must also be given.

-p Specifies the id for a defined list of instances to be copied.

The id refers to a specific collection policy based on direct-instance polling. Information about this type of collection policy is stored in the dsi_polling_list table.

-r Specifies copying TRENDit related tables. Valid values are:

r	Raw tables
R	Rate tables
H	Hourly roll-up tables
D	Daily roll-up tables
W	Weekly roll-up tables
M	Monthly roll-up tables
A	All TRENDit roll-up tables

-R Specifies copying TRENDsum related tables. Valid values are:

r	Raw tables
R	Rate tables
H	Hourly roll-up tables
D	Daily roll-up tables
W	Weekly roll-up tables
M	Monthly roll-up tables
Q	Quarterly roll-up tables
Y	Yearly roll-up tables
A	All TRENDsum roll-up tables

-s Identifies the source server.

-S Identifies the target server.

-t Identifies the tables to be copied. Enter the name of the table as shown in the Object Name column of the Data Manager display. (See "Data Manager" on page 8-1 for details.)

-T Identifies the source topology database.

-u Displays a listing of all TRENDcopy options.

-U Specifies the user name.

-v Provides filtering by view. Use of this parameter is optional. It can only be used in conjunction with the -U parameter.

-V Displays the TRENDcopy version.

-w TRENDcopy checks the dbstats tables to determine the current size of the database to be copied and does not run if the database-used size exceeds the percentage specified in this parameter. The default is 90 for 90%.

Usage Notes

Using the -s, -S, and -t Options To Copy Database Tables

The **-S** (target server) option is required. The **-s** (source server) is optional, but we recommend you specify it for documentation purposes. The source server specified by **-s** is the server from which data and table definitions will be read. Its default value is `$DSQUERY`. The **-t** option specifies which tables within the database will be copied. If the **-t** option is not used, those non-system tables listed in dsi_tab_alias will be copied.

On the target server, the entry in dsi_trendalyzer is created with all fields set to default. If the corresponding key table does not exist, it is created and the dsi_key_tables table updated accordingly.

In addition to the standard syntax listed above, the **-t** option can be used in multiple ways. The destination key table and destination table names can be eliminated from the command string as follows:

```
trendcopy -t source_table -s source_server
-S target_server
```

If you want to omit the key table name and retain the destination table name, however, the syntax becomes:

```
trendcopy -t source_table::destination_table
-s source_server -S destination_server
```

When the destination table name is specified and no entry exists for this name, the compression routine creates a table name. If source or destination table names include the key words rate, hour, day, week, or month, the first letter of the created table name is modified to the corresponding roll-up level.

TRENDcopy provides an error message when the *destination_key_table* name of the **-t** option is improper or invalid. It continues the copy process using a default naming convention for the destination key table unless the destination data table and destination key table already exist.

For example, TRENDcopy creates the destination data table called `basic_test_info_` and the destination key table called `Kasic_test_info_` from the following syntax:

```
-t basic_test_info_:ktest_info:basic_test_info_
```

It creates the destination key table name by replacing the first character of data table name with a `K`.

Multiple tables can be specified on a single command line. If, for example, you want to copy the tables mib-II_ifEntry, mib-II_system, and rmon_history from the server PRIMARY_SYBASE to the server BACKUP_SYBASE, the following command is used:

```
trendcopy -t mib-II_ifEntry -t mib-II_system
-t rmon_history -s PRIMARY_SYBASE -S BACKUP_SYBASE
```

Using the -T Option To Specify a Topology Database

TRENDcopy allows you to override the default topology database when you copy data. If you do not specify a topology database, TRENDcopy uses the topology database defined for $DSQUERY. The -T option is only used to specify a different topology database.

For example, assume you want to copy data tables from PRIMARY_SYBASE to BACKUP_SYBASE. PRIMARY_SYBASE has the default topology database **Chicago**; you want to apply the topology database **Dallas**. The command is:

```
trendcopy -s PRIMARY_SYBASE -T dallas -S BACKUP_SYBASE
```

Using the -U Option To Specify a User Name

When you copy data using TRENDcopy, you can specify the defaults that apply to a particular TREND user. If you do not specify a user, TRENDcopy uses the defaults associated with the default group.

For example, assume you want to copy data tables from PRIMARY_SYBASE to BACKUP_SYBASE, but you want to apply the topology database that belongs to the user Joe Bloggs. If Joe Bloggs' user name is joebloggs, the command is:

```
trendcopy -s PRIMARY_SYBASE -U joebloggs -S BACKUP_SYBASE
```

Using the -r and -R Options To Copy Rolled-Up Data

TRENDcopy copies all non-system tables present in dsi_tab_alias when you omit the -t option. Use the -R and/or -r option to select only certain types of these tables for copying.

Use the -R option to select only raw (r), rate (R), hourly (H), daily (D), weekly (W), monthly (M), quarterly (Q), and/or yearly (Y) tables from the TRENDsum family of rollup tables. Except for raw and rate tables, the table names begin with S and the indicated letter. You can combine these values to select more than one type; for example, use -R DW to select all SD and SW tables. The value A indicates all these types: rRHDWMQY.

Use the -r option to select only raw (r), rate (R), hourly (H), daily (D), weekly (W), and/or monthly (M) tables from the TRENDit family of rollup tables. Except for raw tables, the table names begin with the indicated letter. You can combine these values to select more than one type; for example, use -r DM to select all D and M tables. The value A indicates all these types: rRHDWM.

You may use either the -r or -R option to select raw and/or rate tables. In addition, you may use both options at the same time to increase the selection. For example, -r HD -R HD selects all hourly and daily tables.

When you use the -t and -r options, TRENDcopy copies the rolled-up version(s) of the specified table in addition to the table itself. Here, the -r option increases the number of tables copied. For example, assume you want to copy a raw table named **rmon_host** and its hourly and daily versions from PRIMARY_SYBASE to BACKUP_SYBASE. The command is:

```
trendcopy -t rmon_host -s PRIMARY_SYBASE -S BACKUP_SYBASE
-r HD
```

Note: When you use the -r option with the -t option, do not use the r value as it is seldom possible to determine the raw table name from the rolled-up table name.

Using the -c and -C Options To Filter Data by Column

To specify filtering on column values, use the -c option for data table columns and the -C option for key table options. Only one column filter can be specified per command line for each data and key table. The format for column options is:

```
trendcopy -c column_name:exp:value
```

where

column_ name Is the name given in the Full Column Name column in the Data Manager Column Info window for the table being copied. (See “Data Manager” on page 8-1 for details.)

exp Is one of the following operators:

=	equal
!=	not equal
<=	less than or equal
>=	greater than or equal
<	less than
>	greater than

For example, assume you want to copy table router1. You want only keys corresponding to table_key 1_ALL and only rows where column InPackets is greater than 300. Use the following command:

```
trendcopy -t router1 -s PRIMARY_SYBASE -S BACKUP_SYBASE
-c InPackets:>:300 -C table_key:=:1_ALL
```

Using the -e, -f, and -g Options To Filter Data by Date

You can use TRENDcopy to copy data filtered by date. The following examples explain the use of the -e, -f, and -g options for filtering by date, day of week, and hour of day. You can specify more than one filter; each filter can be a single value or a range of values.

Example 1

The following command copies any data for which the ta_period is 1:00 p.m. on either Monday or Friday:

```
trendcopy -f mo -f fr -g 13 -s xyz -S XYZ
```

Example 2:

The following command copies data with ta_period greater than or equal to Monday and less than or equal to Wednesday; and ta_period greater than or equal to October 1, 1998, and less than or equal to November 1, 1998:

```
trendcopy -f mo:we -e 19981001:19981101 -s xyz -S XYZ
```

Note: The weekday range used with the -f option is Sunday through Saturday, and does not wrap around to the following week. To specify a day range that wraps around to the following week, two command-line options are required (e.g., trendcopy -f we:sa -f su:mo -s xyz -S XYZ will copy data from a Wednesday through the following Monday.)

Example 3:

The following command copies data with ta_period greater than or equal to 1 a.m. and less than or equal to 3 a.m.; and ta_period greater than or equal to October 1, 1998, and less than or equal to November 1, 1998:

```
trendcopy -g 1:3 -e 19981001:19981101 -s xyz -S XYZ
```

Note: The hour-of-day range used with the -g option is 1 through 24, and does not wrap around to the following day. To specify an hour range that wraps around to the following day, two command-line options are required (e.g., trendcopy -g 20:24 -g 1:7 -s xyz -S XYZ will copy data from 10:00 p.m. to 7:00 a.m.)

Using the -v and -n Options To Filter by View and by Type

TRENDcopy allows you to copy information about the nodes contained in a view or a type. If you do not specify either a view or a type, or both, TRENDcopy uses the nodes contained in the topology database of the default group. The -T option is only used to specify a *different* topology database.

For example, assume you have a view named **alabama** containing all of the nodes on your network that are located in the state of Alabama. You want to copy the data concerning this view from OLYMPUS_SYBASE to VESUVIUS_SYBASE. Use the following command:

```
trendcopy -s OLYMPUS_SYBASE -v alabama -S VESUVIUS_SYBASE
```

Using the -b and -l Options To Copy Data

TRENDcopy provides two parameters, -b and -l (lowercase letter l), that, used in tandem, allow you to copy data from a specific baseline period in time. The -b option specifies a baseline, and the -l option specifies the length of the time period to copy.

TRENDcopy establishes the baseline as it relates to the current day at 12:00:00 a.m. The -b parameter defines how many days, weeks, months, or years before today the baseline will be. If today is October 1, and you want to establish the baseline as June 1, you must tell TRENDcopy to set the baseline four months in the past (`-b 4m`).

The -b parameter defines the end of a time period. If today is May 1st, and you wish to copy all the data in your selected tables before this year, set -b at 4 months. TRENDcopy copies all the data stored in the specified tables before the baseline time period.

The -l option can only be used in conjunction with the -b option. The purpose of the -l option is to specify a length of time from the baseline, forming a finite segment of time to copy. If today is February 14, 1996, and you want to copy data from the calendar year 1995, you set the baseline at January 1, 1996 (`-b 45d`) and the length of time at one year (`-l 1y`). TRENDcopy then copies the data from the specified tables for the year that ended at 12:00:00 a.m., January 1, 1996.

Example 1

Assume today is January 11, 1996, and you want to copy 1995 data from PRIMARY_SYBASE to BACKUP_SYBASE. Use the following command:

```
trendcopy -s PRIMARY_SYBASE -b 11d -l 1y -S BACKUP_SYBASE
```

Example 2

Assume today is November 15, and you want to copy information from data tables contained in PRIMARY_SYBASE to BACKUP_SYBASE. The information you want is from May 29 through August 15. Use the following command:

```
trendcopy -s PRIMARY_SYBASE -b 3m -l 2m 18d
-S BACKUP_SYBASE
```

Note: When defining baseline and length, you cannot specify a zero length. In addition, you can only define one baseline, length combination on any one TRENDcopy command.

19 TRENDlabel

TRENDlabel populates one or more columns in a key table with data from its counterpart data table.

Note: TRENDlabel cannot be used with summarized data tables. That is, columns from tables generated by TRENDsum will not populate columns in a key table.

Each data table in the TREND database is associated with a properties (key) table. By default, most of the columns in a data table are not present in its associated key table. However, you may find it useful to include some of these data columns in the key table. TRENDlabel allows you to do this.

TRENDlabel is executed from the command line, and can be scheduled to run regularly in trendtimer.sched.

Synopsis

trend_label [**-c** [*alias*=]*column* ...]

[**-d** *debug_level*]

[**-e** *column*]

[**-h**]

-k *key_table*

[**-o** *hour*]

[**-r** [*alias*=]*column* ...]

-t *source_data_table*

[**-V**]

Options

-c Identifies the column in the source data table that is used to populate the column in the key table. *alias* is the name assigned to the column in the key table and *column* is the name of the column in the source data table. TRENDlabel searches the key table for the column named by *alias*. If the column named *alias* exists, and if its value is null or blanks, TRENDlabel populates the column with data from the column named *column* in the source data table. (If the column exists and its value is nonnull or nonblank, the existing value in the target key table row is not changed.)

If the column named *alias* does not exist in the key table, TRENDlabel creates it and then populates it in the manner described above.

You can repeat this option multiple times in the TRENDlabel statement to identify different key and data table columns or to concatenate column substrings or columns in the source data table to populate the key table column.

If you omit *alias*= from the command, the key table column is assigned the same name as the column in the source data table.

See "Usage Notes" on page 19-4 and "Examples" on page 19-8 for a complete description of this option with examples.

-d Sets the debug output level. Valid values are 1, 2, and 3. The higher the number, the more detailed the information. Debug output is written to the standard output destination. This option is for development purposes.

-e Identifies the column in the source data table that provides the data to populate the **dsi_descr** column in the key table.

-h Displays the syntax of the **trend_label** command.

-k Identifies the destination key table. Specify the SQL name of the table.

-o This option (lowercase letter o) specifies the hour of the previous day. Valid values are 0 (midnight) through 23. This value is used to locate the row in the source data table associated with the **dsi_key_id** that is used to populate the key table columns.

If this option is omitted, the default target row in the source data table is the one where the maximum **ta_period** that is less than or equal to the current time (the minutes and seconds portions of the **ta_period** value are set to zeroes) minus 1 hour on today's date.

See "Locating the Target Row in the Source Data Table" on page 19-5 for a detailed explanation of how the target row is located.

-r Has the same syntax and application as the **-c** option with the following exception: The target value from the source data table populates the associated column in the key table even if that column already exists and has a nonnull or nonblank value.

-t Identifies the SQL name of the source data table. The table is typically a rate table (but can be a raw table). Summary tables cannot be used as a source.

-V Displays the current TRENDlabel command version. For example, a return value of 3.5 means TREND version 3.5.

Usage Notes

These notes provide information on various aspects of TRENDlabel command usage.

Note that omitting *alias*= from the command may produce an ambiguous error message such as `<prog-name> <date/time> - Ambiguous column name XXXX`, which indicates a situation where the key table and data table both have a column called XXXX. The query uses both key and data tables in the *from* clause and uses only XXXX in the *select* clause; it does not specify the owner table name.

To resolve this situation, do the following:

1. Start an ISQL session from a command line with user **`dsi_dpipe`**.
2. Perform **`sp_help`** on the key table to get column-name listing.
3. Drop the improper column from the key table with the following syntax:

   `alter table` <*tablename*> **`drop`** <*column-name*>

4. Exit the ISQL session.
5. Correct the TRENDlabel syntax to use *alias*=*column*.

Populating the dsi_descr Column

When you generate a TRENDgraph or Grade of Service report, the report title appears above the graph and the legend appears below the graph. If the value of the **dsi_descr** column in the key table the report uses is null or blank, the legend simply displays the name of the node on which the report is based. However, if the **dsi_descr** column is populated, the legend displays the description that appears in the column. This can make graphical reports easier to identify and more comprehensible.

Locating the Target Row in the Source Data Table

The source data table contains one row for each unique **dsi_key_id** for each collection period. Thus, if collections are made in 15 minute intervals, there are 96 rows for each unique **dsi_key_id** (four rows per hour * 24 hours per day = 96 rows).

If the **-o** option is used, the hour (0-23) specified in the option is compared to the **ta_period** timestamp in rows of the source data table as follows:

1. Get the timestamp for the current time minus 24 hours. Thus, if the current time is 11:22:35 on June 12, 1998, the resulting timestamp is 11:22:35 on June 11.
2. Make the hour portion of the timestamp equal to the value (0-23) specified in the **-o** option. Thus, if **-o 4** is specified on the TRENDlabel command line, the timestamp is 04:22:35 on June 11.
3. Set the minutes and seconds portions of the timestamp to zeroes. In this example, 04:22:35 on June 11 becomes 04:00:00 on June 11. Thus, the target record in the source data table in this example is the one for the **dsi_key_id** where **ta_period** is Jun 11 1998 04:00:00.
4. If there is no row with the target time period, the hour is decremented by 1 until a match is found. For example, if there is no row for the **dsi_key_id** where the **ta_period** time is 04:00:00, TRENDlabel looks for a time of 03:00:00, decrementing the hour by 1 in this manner until a match is found.

If the **-o** option is omitted, the target row in the source table is the one for **dsi_key_id** where the maximum ta_period timestamp is less than or equal to the current time (with hours and minutes zeroed) minus 1 hour. For example, if the current time is 10:30 AM on June 12, 1998:

1. Get the **dsi_key_id** record in the source data table where the **ta_period** timestamp is Jun 12 1998 09:00:00 AM.
2. If no record exists for 09:00:00, TRENDlabel decrements the hour by 1 until a match is found. For example, if there is no row for the **dsi_key_id** where the **ta_period** time is 09:00:00, TRENDlabel looks for a time of 08:00:00, decrementing the hour by 1 in this manner until a match is found.

Ensuring Key Table/Data Table Compatibility

You must ensure that the specified data table (**-t** option) uses the specified key table (**-k** option) in the TREND database. TRENDlabel does not check for this.

Update Restrictions

You cannot update the values in the key table columns **dsi_key_id** **dsi_target_name**, or **dsi_table_key**.

Extracting Substrings from Column Values

When you use the **-e**, **-c**, and **-r** options, you can specify that a substring be extracted from the source column and used to populate the destination key table column with the following syntax:

*column***:***offset***,***length*

where:

column Is the column name in the source data table.

offset Is the starting character position of the substring in the column.

length Is the number of characters to include in the substring beginning with the offset column position.

For example:

```
ta_period:13,8
```

means that the 8-character substring beginning in position 13 of **ta_period** be extracted from the column value.

Concatenating Column Values

You can populate a key table column with a value that is concatenated from the values of multiple columns and/or column substrings in the source data table by repeating the **-e**, **-c**, and **-r** options using the same alias= and different source column names or substrings. For example, the following TRENDlabel command:

```
trend_label -k Ksi_dbstats_ -t Rsi_dbstats_
-c dsi_eurodate=ta_period:5,3 -c dsi_eurodate=ta_period:1,4
-c dsi_eurodate=ta_period:8,4
```

creates (if needed) and populates a key table column named **dsi_eurodate** with the concatenated substrings from positions 5-7, 1-4, and 8-11 of the **ta_period** column in the source data table. In this case, if the value of **ta_period** in the source data table is:

```
Jul 29 1998 04:00:00:000AM
```

the value used to populate **dsi_eurodate** in the key table is:

```
29 Jul 1998
```

Examples

Example 1:

The TRENDlabel command:

```
trend_label -k Kib_ii_ifentry_ -t mib_ii_ifentry_ -e ifdescr010
```

populates the **dsi_descr** column in the Kib_ii_ifentry_ key table with the data contained in the **ifdescr010** column from the mib_ii_ifentry_ source data table. If the time when the **trend_label** command executes is 11:45 AM, the **dsi_key_id** record with a **ta_period** value of today's date at 10:00:00 AM is used. If there is no record for 10:00:00 AM, TRENDlabel searches for a record for 09:00:00 AM and so on backward in 1-hour increments until a match is found (the maximum **ta_period** value that is equal to or less than the current hour minus 1 hour).

Example 2:

The TRENDlabel command:

```
trend_label -k Kib_ii_ifentry_ -t mib_ii_ifentry_ -e ifdescr010
-c speed=ifspeed013 -c type=iftype011
```

populates the **dsi_descr** column in the Kib_ii_ifentry key table the same way described in example 1.

In addition, it creates (if the column does not already exist) and populates the key table column named **speed** with the value for **ifspeed013** in the target source data table row and the key table column named **type** with the value for **iftype011** in the target source data table row if the key table column values are null or blank. If the value in either column is nonnull or nonblank, the current value is not replaced.

The target source row is identified the same way as described in example 1.

Example 3

The command:

```
trend_label -k Ksi_dbstats_ -t Rsi_dbstats_ -e user_name -e
ta_period:1,3
```

populates the **dsi_descr** column in the Ksi_dbstats key table with a concatenation of the value for the **user_name** column and the first three characters of the **ta_period** value in the target source table row. Thus, if the **user_name** value is **Jones** and the first three characters of the **ta_period** value are **Nov**, the value **JonesNov** is used as the **dsi_descr** column value in Ksi_dbstats (if the existing value is null or blanks). If the existing **dsi_descr** value is nonnull or nonblank, the value is not changed.

The target source row is identified the same way as described in example 1.

Example 4

The command:

```
trend_label -k Ksi_dbstats_ -t Rsi_dbstats_ -o 4
-r dsi_descr=user_name -r dsi_descr=ta_period:1,3
```

populates the **dsi_descr** column in the Ksi_dbstats key table with a concatenation of the value for the **user_name** column and the first three characters of the **ta_period** value in the target source table row the same ways as described for example 3 with the exception that the existing value of **dsi_descr** is replaced by the concatenation even if that value is nonnull or nonblank. The **-r** option (used in example 4) causes unconditional replacement; the **-e** option (used in example 3) replaces only if the existing value in the target key table record is null or blanks.

Furthermore, the target record in the source data table is the **dsi_key_id** record where the **ta_period** value is 04:00:00AM on the previous day.

Example 5

The command:

```
trend_label -k Ksi_dbstats_ -t Rsi_dbstats_ -o 4
-e user_name -e ta_period:1,3
-r dsi_eurodate=ta_period:5,3 -r dsi_eurodate=ta_period:1,4
-r dsi_eurodate=ta_period:8,4
```

takes data from Rsi_dbstats where **ta_period** for the dsi_key_id equals 04:00:00 AM on the previous day. It concatenates **user_name** and the first three characters of the **ta_period** value in Rsi_dbstats_ and places the result in the **dsi_descr** column in Ksi_dbstats_. It creates a column named **dsi_eurodate** in Ksi_dbstats_ (if the column does not already exist), and unconditionally updates that column with the result of concatenating positions 5-7, 1-4, and 8-11 in **ta_period**.

20 TRENDexporter

This chapter describes the TRENDexporter utility. This application allows you to run a TRENDsheet report against your database and to store or export the output in an ASCII text format.

The document has four parts:

- "Overall Process and Functionality" on page 20-1.
- "Controls and Switches" on page 20-3.
- "Input Parameter File" on page 20-6.
- "Output Format" on page 20-14.

Overall Process and Functionality

trend_export generates ASCII files of data extracted from TREND using the SQL query defined in any TRENDsheet query file. The ASCII file generated can be saved in any location. In short, here is the process involved:

1. Generate a source TRENDsheet query file using TRENDbuild and TRENDsheet.
2. Invoke the ASCII generator utility, pointing to this source query file.

The utility makes use of the SQL generated by TRENDbuild and TRENDsheet and saved in a query file with a .qss file name extension. The utility uses the source SQL in the file - including the constraints - as the basis for a database query, and stores the rows returned as an ASCII file. The query can be limited to a single device/instance pair or run against a set of pairs. The output can be saved in a separate file for each pair or all the data can be combined into a single output file. trend_export can be invoked on a single TRENDsheet report or for a set of reports by using an input parameter file.

The ASCII files generated can be in either a tab separated format or a comma separated values (CSV) format. The files can contain or omit column headers. A facility for generating data for just the most recent time period or between to specific dates is also available. The number of lines to be processed can be limited, which prevents the generation of overly large output files.

Lastly, each report to be run, either individually from the command line or through a parameter file, can query any SQL Server system accessible from the user’s machine.

Controls and Switches

There are multiple parameters to control the execution of the program. The command line options are:

Flag	Description	Argument	
-a	Include column headers. Headers include both a line for the column names and a line of hyphens (-) underlining each column name.	y	The ASCII file will include column headers (DEFAULT).
		n	The ASCII file will not include column headers.
-c	Condense the resulting ASCII output file. In an uncondensed file, each column is padded with trailing spaces up to its default width from its database definition. A condensed file has these trailing spaces removed.	y	Condense the ASCII file.
		n	Do not condense the ASCII file (DEFAULT).
-d	Show debug output. If this option is included, the entries made in the log file are also displayed on your screen. This option should only be used for testing in coordination with DeskTalk Systems Technical Support due to the additional overhead it places on rmon_collect.	0	No debug output.
		1, 2, or 3	The higher the number, the more detailed the debug output.

(1 of 3)

Flag	Description	Argument	
-f	Specifies the format of the ASCII output file.	tab	Generate a tab separated ASCII output file (DEFAULT).
		comma	Generate a comma separated ASCII output file. Character data is enclosed in quotes.
-h	Help. Display command line options.	N/A	
-i	The full path and file name of the input parameter file.	The name and directory of any valid parameter file.	
-m	Only include data with the most recent timestamp. (i.e. the maximum ta_period value in the source data table).	y	Only include data with the most recent timestamp.
		n	Include all data regardless of timestamp (DEFAULT).
-o	The name of the ASCII output file. To insure that existing files aren't overwritten, the actual output file name will have a timestamp appended to it in the format YYYYMMDDHHMM. If -o myfile.txt is specified and trend_export is run on April 10, 1997 at 3:30pm, the output file name will be myfile.txt.19970410153.	The complete path name of the ASCII output file to be generated.	

(2 of 3)

Flag	Description	Argument
-p	The maximum number of lines to be generated in the output file.	Any positive integer (DEFAULT = 5000).
-q	The name of the SQL Server from which the data will be extracted.	The name of the SQL Server. (DEFAULT = Current value of the user's DSQUERY environment variable.)
-r	The name of the TRENDsheet (.qss) query to be used.	The complete path name of any TRENDsheet query file.

(3 of 3)

Note that whenever the -i option is used, all other command line options are ignored except the -q option. In this case, the named input parameter file controls ASCII file generation exclusively.

The utility needs one environment variable set: DPIPE_HOME. This variable contains the path name of the directory where all TREND system components reside. The utility writes all of its messages to the trend.log file and looks for the export control parameter files in DPIPE_HOME/lib.

Here are a few examples of launching trend_export from the command line:

```
trend_export -r /trendsnmp/reports/Daily_Lan_Inventory.qss -o /
daily_lan_report.txt -f comma
```

This would use the TRENDsheet report query file Daily_Lan_Interface_Inventory.qss as input and produce an output ASCII file with the name daily_lan_interface_inventory.txt. Notice that the fully qualified path names were specified for both the report and the ASCII output file. In addition, note the use of the -f option specifying that the ASCII file should be comma delimited.

```
trend_export -i /trendsnmp/reports/ascii_parm.dat
```

This invocation would use the input parameter file ascii_parm.dat found in the directory named $DPIPE_HOME\lib to drive the ASCII file creation process. See "Input Parameter File" on page 20-6 for the parameter file format.

trend_export -i /trendsnmp/reports/ascii_parm.dat -q hidekel

This invocation would work like the previous one but would use the SQL server hidekel as the source of the data generated during processing.

Input Parameter File

When using the -i option to point the process to an input parameter file all other command line options are ignored except for -q (DSQUERY). This is the preferred method for using the utility. Only by using the input parameter file option can multiple reports and multiple instances of those reports be generated. Here are the guidelines for the parameter file:

The rules for formatting a parameter file are:

1. One entry per line.
2. All entries of keywords with arguments use the format <keyword> = <argument>.
3. Each report to be run must be blocked off with a start and end statement.
4. An * in the first column indicates a comment.
5. Any keywords not listed here will be ignored.

Valid keywords for the parameter file are:

Keyword	Associated Command Line Option	Required	Valid Values	Default
start		Yes	N/A	N/A
end		Yes	N/A	N/A
report	-r	Yes	Any qss file path name	None
output	-o	Yes	ASCII file path name	None
headers	-a	No	y n	y (yes)
format	-f	No	comma tab	tab
dsquery	-q	No	valid SQL Server name	DSQUERY environment variable value
max	-m	No	y n	n (no)
exception		No	y n	n (no)
instances		No	valid target name/table key pairs, or file containing target name/table key pairs. See below for format.	None

(1 of 2)

Keyword	Associated Command Line Option	Required	Valid Values	Default
maxlines	-p	No	any positive integer	5000
condense	-c	No	y n	n (no)
style		No	individual combine	individual
begindate		No	date in US format	None
begintime		No	timestamp hh:mm	00:00
finishdate		No	date in US format	None
finishtime		No	timestamp hh:mm	23:59
frequency		No	daily, weekly, monthly	None
*		No	Any text. This is a comment line.	

(2 of 2)

The format for an instances line is:

```
instances =
(target_name,table_key);(target_name,table_key)... ..
```

Multiple instances lines can be added for any report (see examples).

```
instances = /tmp/exception_list
```

This usage of the keyword points to a file containing a list of target name/table key pairs in the form:

```
target-name1,table-key.1
target-name1,table-key.2
target-name2,table-key.1
etc... ..
```

Note that the instances file can be generated by specifying a report as an exception report. When a report specification block contains the exception=y keyword, the ASCII output file generated is simply a list, in the above described format, which is the matching target name/table key pairs from the exception criteria. This allows for the results from an exception report to be used to drive drilldown reports. See example 4 below.

The style keyword determines whether the rows returned for multiple instances are combined into a single output file written to separate files, one for each target name/ table key. If no instances are listed the style keyword is ignored. If style = individual is specified, each output file name will have <target_name>.<table_key> appended to its name in addition to the timestamp described above.

The condense keyword causes trend_export to remove trailing blanks from each column included in the SQL select statement.

If the max keyword is used, then the begindate, finishdate, begintime and finishtime constraint keywords are ignored. Also, any imbedded start or end date constraints in the original .qss file will be ignored.

If the frequency keyword is used (daily, weekly or monthly) then any other date constraint is ignored. That is, if in the same report specification block the max keyword and/or the begindate and/or the finishdate keywords are used, then they will be bypassed.

If you plan on running the same report on a daily, weekly or monthly basis, then use the frequency keyword for the required time period. That way, no modification to the control parameter file will be needed. By specifying the daily, weekly or monthly frequency value, the following actions are done:

for daily =>	the previous day's data is queried for. For example, if the current date is 6/1/97, then the report would query for data from 5/31/97 00:00 to 5/31/97 23:59. No matter when the report is run, the previous day's data is queried for.
for weekly =>	the previous week's data is queried for. For example, if today was Wednesday May 28th, 1997, then the report would query for data from Monday May 19th, 1997 00:00 to Sunday May 25th, 1997 23:59. Again, no matter when the report is actually run, the previous week's data is queried for.
for monthly =>	the previous month's data is queried for. For example, if today was 5/7/97, then the report would query for data from April 1st, 1997 00:00 to April 30th, 1997 23:59.

Example 1:

```
* this is a comment line
*
start
report = /apps/trendsnmp/reports/Mib-II/
Daily_LAN_Interface_Inventory.qss
output = /tmp/daily_lan_interface_inventory
headers = n
format = comma
dsquery = HARDWARE_SYBASE
condense = n
maxlines = 2000
style = individual
end
```

Example 1 would run the TRENDsheet query file Daily_LAN_Interface_Inventory.qss and place the output in /tmp/daily_lan_interface_inventory.<timestamp>. The ASCII file generated would have no column headers and would be in a condensed, comma delimited format. Lastly, the query would be run against the SQL server HARDWARE_SYBASE.

Example 2:

```
* this is a comment line*
start
report = /apps/trendsnmp/reports/Mib-II/
Daily_LAN_Interface_Inventory.qss
output = /tmp/daily_lan_interface_inventory
headers = n
format = comma
dsquery = HARDWARE_SYBASE
end

start
report = /apps/trendsnmp/reports/Mib-II/
Serial_Interface_Usage.qss
output = /tmp/serial_interface_usageendstart
report = /myreports/lan_interface_load.qss
output = /myreports/lan_interface_load
max = y
instances = (mynode1,1);(mynode1,2);(mynode2,1);(mynode2,2)
instances = (mynode3,3);(mynode4,1)
end

start
report = /myreports/lan_interface_load.qss
output = /myreports/lan_interface_load
max = y
instances = (mynode1,1);(mynode1,2);(mynode2,1);(mynode2,2)
instances = (mynode3,3);(mynode4,1)
style = combine
format = comma
condense = y
end
```

Example 2 causes trend_export to run a set of reports.

1. First it would run the same report as in Example 1.
2. The second report run would be Serial_interface_Usage.qss. Its output would be a tab separated, uncondensed data saved in a file called /tmp/

serial_interface_usage.<timestamp>. Notice that all the default values would be used for this second report.

3. The third report to be run, lan_interface_load.qss would produce six tab separated, uncondensed ASCII files because the keyword style = individual is used. The qss report source file is used as a template to run the query for each of the listed instances. Note that the instances are on multiple lines. They could have been all listed on the same line. Also note that the max keyword was added to indicate that the ASCII files will only contain data for the most recent period available for each instance. The output files will be named:

   lan_interface_load.mynode1.1.<timestamp>,
   lan_interface_load.mynode1.2.<timestamp>,

   lan_interface_load.mynode2.1.<timestamp>,
   lan_interface_load.mynode2.2.<timestamp>,

   lan_interface_load.mynode3.3.<timestamp>,
   lan_interface_load.mynode4.1.<timestamp>.

4. The fourth report would produce the same data as the third report, but it would only produce a single output file called lan_interface_load.<timestamp>. In addition, the output file would be comma delimited and condensed.

Example 3:

```
start
report = /myreports/lan_interface_load.qss
output = /myreports/lan_interface_load
max = n
begindate = 03/01/97
finishdate = 03/31/97
instances = (mynode1,1);(mynode1,2);(mynode2,1);(mynode2,2)
instances = (mynode3,3);(mynode4,1)
style = combine
format = comma
condense = y
end
```

Example 3 causes trend_export to run a set of reports, in a combined format (i.e., all the data from the listed instances is combined into one output file) for only data whose ta_period lies between March 1st and March 31st.

Example 4:

```
start
report = /myreports/critical_interfaces.qss
output = /tmp/critical_interfaces_list
frequency = daily
exception = y
max = n
begindate = 03/01/97
finishdate = 03/31/97
format = comma
end
start
report = /myreports/lan_interface_performance_report.qss
output = /myreports/heavy_hitters
frequency = daily
max = n
instances = /tmp/critical_interfaces_list
format = comma
style = individual
end
```

Example 4 demonstrates the use of the exception keyword. In the first report the exception keyword has been turned on. This causes the system to generate a target name/table key listing in the file /tmp/critical_interfaces_list'. The generation of this list is based upon whatever criteria was specified in the exception report. The output from this exception report is then used as an instances input list for the next report. Note that the second report is set to style = individual, which means that for each target name/table key pair a separate output file will be generated. Also note that the frequency = daily keyword was used. This means that both the exception report and the drilldown report will only query for the previous day's data.

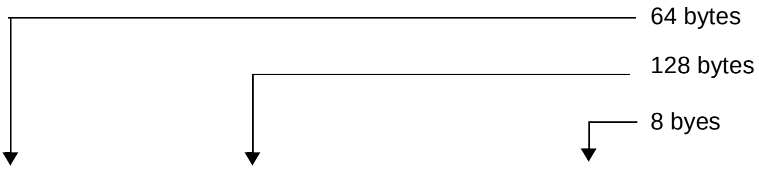
Output Format

As stated earlier, the ASCII files can be generated in two formats, tab or comma separated and both formats can be generated either in a condensed or uncondensed mode. Here are examples of how the output would look in the various possible combinations of these methods. They are based on a TRENDsheet report which utilized the following columns:

Column Name	**Size in Database**
dsi_target_name	64 bytes
dsi_table_key	128 bytes
ifinoctets018	8 bytes

Tab Separated Format: Not Condensed

The tab format consists of columns of data aligned with the next in a series of standard, equally spaced tab stops. That is, the actual amount of separation between columns is the number of spaces from the end of the column to the next tab stop. Because this example is not condensed, the amount of space occupied by each column is based on the field size defined in the database. Below is an example of a tab separated ASCII output file with the headers option set to yes:



```
dsi_target_name    dsi_table_key             ifinoctets018
your-router        1                         123456
your-router        2                         123456
your-router2       1                         1234567
```

The output generated if the headers = n keyword was specified would be the same data in the same format but without the column headings:

```
your-router        1                         123456
your-router        2                         123456
your-router2       1                         1234567
```

Comma Format: Not Condensed

The comma format consists of columns of data separated by commas. Each column occupies the number of bytes specified for it in the database. Note that whenever a comma delimited file is requested, all character based data is enclosed in quotes. With headers = y set, the output would look like this:

```
dsi_target_name    ,dsi_table_key            ,ifinoctets018
’your-router’      ,’1’                      ,123456
’your-router’      ,’2’                      ,123456
’your-router2’     ,’1’                      ,1234567
```

Without headers, the same data would be generated, but without the column headings:

```
’your-router’      ,’1’                      ,123456
’your-router’      ,’2’                      ,123456
’your-router2’     ,’1’                      ,1234567
```

Tab Format: Condensed

The condensed mode of the tab separated format generates a tab delimited file, but the columns are only as wide as the actual data rather than being padded to the defined field width. The columns are aligned with the next tab stop. With headers = y the output would be:

```
dsi_target_name dsi_table_key ifinoctets018
your-router 1 123456
your-router 2 123456
your-router2 1 1234567
```

This same output with headers=n would be:

```
your-router 1 123456
your-router 2 123456
your-router2 1 1234567
```

Comma Format: Condensed

The condensed comma format is a series of columns that are only as wide as the actual data they contain, separated by commas. Note that whenever a comma delimited file is requested, all character based data is enclosed in quotes.

With headers, the ASCII file generated would be:

```
dsi_target_name,dsi_table_key,ifinoctets018
'your-router','1',123456
'your-router','2',123456
'your-router2','1',1234567
```

If the headers=no option was used, then this would appear as:

```
'your-router','1',123456
'your-router','2',123456
'your-router2','1',1234567
```

21 TRENDproc

TRENDproc is a utility that groups together multiple interrelated commands. This allows the TREND user to chain executable commands together, allowing them to be run sequentially, at a predetermined time. In addition, TRENDproc enables the execution of multiple instances of an application, allowing them to run in parallel.

How TRENDproc Works

The concept behind TRENDproc is relatively simple:

1. Take a sequence of processes, which, in the past, had to be accomplished individually, and string them together in such a way that they can be executed with a single command.

   In TRENDproc terminology, this series of sequential commands makes up a block—the equivalent of a macro.

2. Next, make it possible to string blocks together like individual commands.

When one block finishes its assigned tasks, automatically launch the next block, and so on until all of the blocks of processes have been executed.

3. Then, to make the process really powerful, make it possible to launch multiple blocks concurrently, so that they run in parallel.
4. Finally, make it possible to launch the entire process with a single entry in trendtimer.sched.

The result is a system that allows TREND users to automate many of the tasks that have traditionally been accomplished manually.

This system is made up of two parts:

- The TRENDproc Application, which reads the trend_proc file, and runs the commands it contains.
- trend_proc files, which contain the commands that are to be run.

The TRENDproc Application

TRENDproc is a program that is initiated by lpr_launch. When launched, TRENDproc reads the appropriate trend_proc file, and executes the scheduled commands in the appropriate order. The TRENDproc is not intuitive; it acts as a robot, following the instructions provided by the trend_proc file.

Since a trend_proc file can itself call another trend_proc file, multiple instances of TRENDproc can run concurrently.

trend_proc Files

A trend_proc file contains one or more blocks of commands. A block is a sequential list of commands to execute. Assuming there are six items in a block, TRENDproc will execute the first item, and wait for that command to be completed. As soon as the first item is complete, TRENDproc initiates the second item. When the second item finishes, TRENDproc starts the third, and so on until all of the items in the block have been executed.

Most trend_proc files contain multiple blocks, arranged in a linear progression. It is necessary for the blocks to be in a particular order, because TRENDproc reads the blocks sequentially.

However, TRENDproc is able to execute commands from different blocks concurrently. This is so because TRENDproc recognizes two different kinds of blocks in a trend_proc file: wait and nowait.

A wait block is so called because it causes TRENDproc to wait until the block has been executed in full before moving on to another block.

A nowait block permits TRENDproc to go to the next block in the sequence, and begin executing the commands in that block.

Example:

A trend_proc file contains eight blocks of commands, three of which are wait blocks. The sequence of the blocks is described below:

Block 1: six commands (wait)
Block 2: seven commands (nowait)
Block 3: four commands (nowait)
Block 4: eleven commands (wait)
Block 5: eight commands (nowait)
Block 6: five commands (nowait)
Block 7: three commands (wait)
Block 8: twenty-seven commands (nowait)

Note: Each block must have a unique name. Since multiple instances of TRENDproc can run in parallel, block names should not be duplicated. The names used above are representative of a naming pattern. If you

intend to create your own trend_proc files, it is a good idea to establish a naming convention for your trend_proc blocks.

This trend_proc file would require four steps to accomplish:

Step 1

TRENDproc is launched (by lpr_launch, based on the appropriate TRENDproc entry in trendtimer.sched), and executes the first command listed in block 1. TRENDproc cannot execute the second command in block 1 until the first command has been completed. Since this block is a wait block, TRENDproc must complete the entire sequence of commands in block 1 before beginning the sequence contained in block 2.

Step 2

TRENDproc executes the final command in block 1, and begins immediately to execute the commands in block 2. Again, TRENDproc must execute the commands in block 2 in their proper sequence. However, since block 2 is a nowait block, TRENDproc also begins to execute the sequence of commands in block 3. Since block 3 is also a nowait block, TRENDproc can also begin to execute the commands in block 4. At this point, TRENDproc is executing commands from three different blocks at the same time.

Block 4 is a wait block, so once again, TRENDproc must complete the commands in block 4 before beginning the processes in block 5. However, it is possible that TRENDproc will execute all of the commands in block 4 before blocks 2 and 3 are completed. In this circumstance, TRENDproc does not have to wait for blocks 2 and 3-once block 4 has been executed, TRENDproc will start with block 5.

Step 3

TRENDproc completes the sequence of commands contained in blocks 2-4 and begins to execute the processes in blocks 5-7. Again, since block 7 is a wait block, the commands in block 7 must be fully executed before TRENDproc can begin the sequence of commands in block 8.

Step 4

When TRENDproc executes the commands in block 8, the sequence is finished.

TRENDproc can execute commands from several blocks at once. In the example above, we see TRENDproc execute commands from three different blocks. This allows a large part of the individual trend_proc file to be run in parallel.

However, TRENDproc allows an even more sophisticated way to run commands in parallel. A trend_proc file can call one or more additional trend_proc files. In fact, multiple instances of TRENDproc can run concurrently, each running parallel processes. The syntax for this call is:

```
trend_proc -f <trend_proc filename >
```

This process of creating a trend_proc that contains one or more trend_proc files can be especially useful. Since it is possible for trend_proc files to conflict with each other, it is sometimes necessary to ensure that one TRENDproc is complete before another is launched. If, however, you create a trend_proc file whose blocks are trend_proc files, you can launch an entire sequence of trend_procs with one entry in trendtimer.sched.

Note: While it is often a good idea to have a master TRENDproc controlling your trend_proc files, you should limit this nesting to two levels. Otherwise, you increase the chance of creating a conflict between your various processes.

Example:

You have created five different trend_proc files (Aspen, Birch, Casuarina, Dogwood, and Elm), each of which performs the data collection, data aggregation and summarization, and report generation for a different set of reports. Since some of these trend_procs utilize data from the same raw data tables, it is necessary for these TRENDprocs to be run one at a time.

You ***could*** create a separate entry in trendtimer.sched for each trend_proc file, provided you spaced them out over time to ensure that each TRENDproc finished before the next was launched.

Rather than this, you create a sixth trend_proc file (Trees) to contain the other five. Its blocks would be as follows:

Block A: Aspen (wait)
Block B: Birch (wait)
Block C: Casuarina (wait)
Block D: Dogwood (wait)
Block E: Elm (wait)

You then put an entry into trendtimer.sched to run Trees once a day.

Possible Applications of TRENDproc

TRENDproc was devised to act as a framework for the new TREND ReportPacks. However, it can be applied in a wide variety of circumstances.

Example 1:

TREND v3.4 uses the TRENDsum application to maximize the efficiency of data aggregation. Since the TRENDsum application requires database tables that are created by trendit, it is necessary to ensure that the trendit process has finished before initiating TRENDsum.

By setting trendit and TRENDsum to run as part of a trend_proc block, you can ensure that trendit has completed its run before TRENDsum kicks off. Such a trend_proc block might look like the one below:

```
begin: block1 wait
trendit -t mib_ii_ifentry_ -r
trend_sum -e Rib_ii_ifentry_ -e SDifentry  -f trendsum.sum
end: block1
```

Example 2:

Those TREND users whose TREND networks include Satellite Servers must export data to their central database every day. Typically, the export process is scheduled separately from the data roll-up.

If you wanted to export the data as soon as it was aggregated, you could put both the roll-up and export processes in a single TRENDproc file. This way, as soon as the data had been aggregated, TRENDproc would automatically export the data to the proper database.

Creating and Applying a TRENDproc

Creating your own TRENDproc files, is a three-step process. You must:

1. Define the commands you want to execute, and in what order they should be run.
2. Create the TRENDproc file.

   You need to make sure you define your blocks in the correct order using the wait and nowait options. Otherwise, things may not run as you need them to.
3. Schedule TRENDproc to launch.

The remaining sections in this chapter describe this process.

Defining a TRENDproc

You can design a TRENDproc to accomplish nearly any sequence of TREND processes. Simply decide which executable files you wish to call, and in which order they should be executed. If you are creating a complex TRENDproc file, you might want to approach it in a hierarchical fashion.

Example:

First Level trendit runs against a specified set of data tables.

If you are aggregating data from multiple databases, you can set one instance of trendit to run against each database, with each instance of trendit executed from a different TRENDproc block.

Second Level TRENDsum runs against the data tables created by trendit.

Again, you can create multiple blocks, each launching one TRENDsum against one set of data tables.

Third Level TRENDexporter exports the aggregated data to the primary TREND database.

Fourth Level A series of reports is run against the new data.

Having established more or less what you want to accomplish, you can define your TRENDproc file to execute the proper commands. In this case, you might create a single block for each (satellite) database you wanted to run trendit and TRENDsum against. Each block would have three commands:

1. Run trendit
2. Run TRENDsum
3. Run TRENDexporter

If your TREND network had six satellite servers, this would mean creating six separate nowait blocks, one for each server. You would then create a seventh block, to run the desired reports against the primary database.

Things to Avoid

You need to avoid trying to access the same database table at the same time. For example, you would not want to have two TRENDsum processes trying to summarize data from the Rib_ii_ifentry_ table at the same time.

TRENDproc File Syntax

A TRENDproc file is a text file, with the extension .pro. It must contain at least one block, and the blocks must adhere to a particular format:

A block begins with statement begin: <blockname>. In addition, this statement defines the wait-nowait condition of the block. The syntax for the beginning of the nowait block pintail is as follows:

```
begin <pintail> nowait
```

Each block must have a unique name.

A block can contain one or more statements, to be executed sequentially. These statements can be:

- The fully qualified path of an executable file
- The command-line execute command of a TREND application

A block does not need to contain a command. In this case, it is an empty block. This will not affect the performance of the TRENDproc-so long as the populated blocks are formatted correctly.

A block ends with the statement:

```
end: <blockname>
```

Note: Putting the block name in the End statement is optional. It can, however, be useful for identification purposes, especially if you nest one TRENDproc inside of another.

- The words begin and end must be in lower case.
- Comments within a block should begin with the number character (#).
- There should not be a space between the begin/end and the colon.
- The default option for wait/nowait is wait. If you do not specify wait or nowait on the begin: line, it will default to wait.

Scheduling TRENDproc

After you've created your TRENDproc file you need to place an entry in the trendtimer.sched file. Otherwise, TRENDproc will not launch at the appropriate time. The syntax of this entry is as follows:

```
trend_proc -f <filename>
```

where

<*filename*> = the name of your TRENDproc file.

22 TRENDdbview

TRENDdbview provides an additional mechanism for viewing and comparing data. It allows you to combine columns from multiple tables and views into a new “virtual” table. Using TRENDdbview, you can compare and contrast specific data from separate tables, either within a single MIB, or from completely different MIBs.

Note: The views created by TRENDdbview are database views. Database views are unrelated to the topology views described in Chapter 2: *TREND Configuration for trendadm*.

This chapter discusses the various aspects of TRENDdbview, what it does, how it works, its components, and how to use it.

Purpose of TRENDdbview

TRENDdbview is used to combine individual columns from one or more data tables, views, and key tables to form a database view. Views are virtual data tables, and

TREND treats them as data tables. The result is that you can select individual columns from several different tables, and bundle them together to create a new table.

Once you have created a view using TRENDdbview, you can use it to generate the same reports as you would for any data table.

How TRENDdbview Works

TRENDdbview collects the columns of data you request from the data tables you specify, and combines them into a view, without actually creating a new table in your database. In the course of using TRENDdbview to define your view, you assign it to a key table.

Each data table has an associated key table. When you use TRENDdbview to select data tables, their key tables also appear. You choose which of these key tables you wish to associate with your view. The view becomes a virtual table, cross-referenced by the key table.

To TREND’s report generating applications, the view appears to be another data table. As a result, you can view the data, and generate and print reports, just like you would for any data table.

Note: Once you create a view in your database, it acts like a data table. This means that you can also use columns from a previously created view as part of a new view, just as if the existing view were a data table.

A Note on Limitations

TRENDdbview does not limit the number of data tables and views that contribute columns to a view. However, TRENDdbview can only support up to two key tables. You can include columns from every member data table in a key table, and you can do so for one or two key tables. However, you cannot combine columns from data tables that are members of three or more key tables.

Elements of TRENDdbview

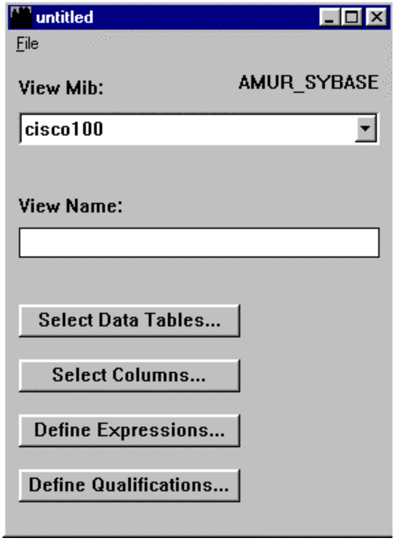
TRENDdbview is made up of five screen elements. These are:

- TRENDdbview main window
- Select Data Tables window
- Define Qualifications window
- Select Columns window
- Define Expressions window

Each of these elements serves a specific purpose, some essential, others optional. The function of each element is described below.

TRENDdbview Main Window

The TRENDdbview main window appears when you launch TRENDdbview.



View Mib: AMUR_SYBASE

cisco100

View Name:

Figure 22-1: TRENDdbview Main Window

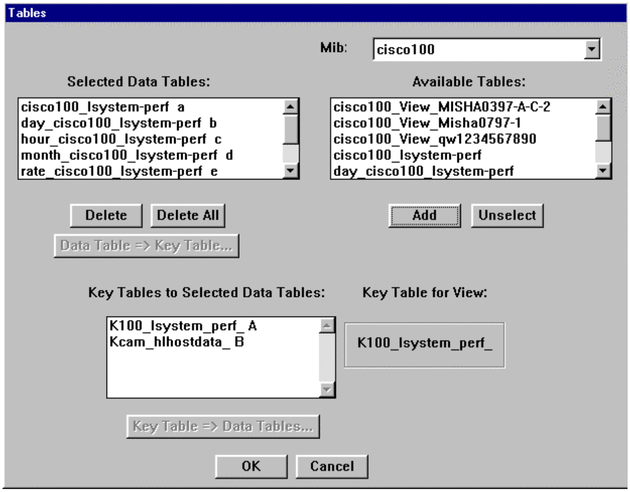
The TRENDdbview main window is the connection to each of the other elements, and is the only path by which they can be reached.

In addition, the TRENDdbview main window contains a File pull-down menu with the New, Open..., Save, Save As..., Run, Change Database..., Exit, and About... commands.

Once you have defined a view, you will use the TRENDdbview main window to determine the name under which the view will be saved. The procedures for accomplishing these actions are described later in this chapter.

Select Data Tables Window

You access the Select Data Tables window by clicking the Select Data Tables button in the TRENDdbview main window. The buttons and fields in the Select Data Tables window allow you to define the data and key tables that make up your view and the key table to be associated with it.



Mib: cisco100

Selected Data Tables:

cisco100_lsystem-perf a
day_cisco100_lsystem-perf b
hour_cisco100_lsystem-perf c
month_cisco100_lsystem-perf d
rate_cisco100_lsystem-perf e

Available Tables:

cisco100_View_MISHA0397-A-C-2
cisco100_View_Misha0797-1
cisco100_View_qw1234567890
cisco100_lsystem-perf
day_cisco100_lsystem-perf

Key Tables to Selected Data Tables:

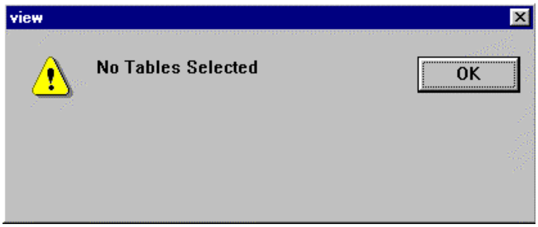
K100_lsystem_perf_ A
Kcam_hlhostdata_ B

Key Table for View: K100_lsystem_perf_

Figure 22-2: Select Data Tables Window

Complete the Select Data Tables window first. Before you can assign qualifications, apply expressions, or choose data columns for a view, you must first define its data and key tables. Each of the other definitions is referenced to the tables you select. For this reason, you must choose your data tables before you can specify columns or assign either expressions or qualifications.

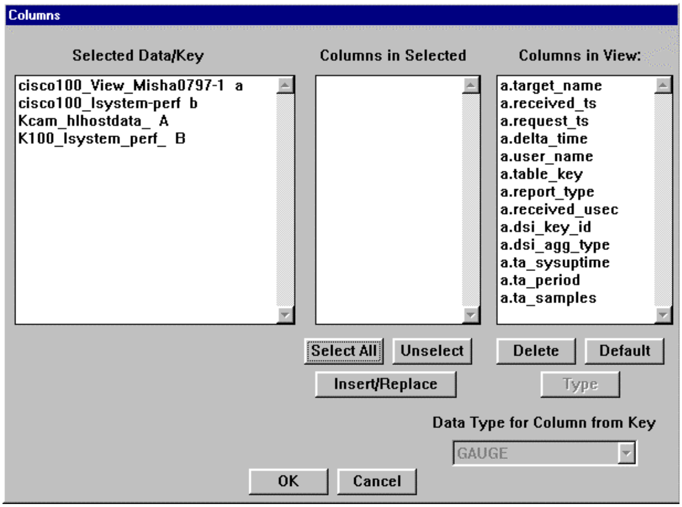
If you try to access another window before you specify at least one data table, the following message appears:



No Tables Selected

Select Columns Window

Use the Select Columns window to specify the data columns to appear in your view.



Selected Data/Key

cisco100_View_Misha0797-1 a
cisco100_lsystem-perf b
Kcam_hlhostdata_ A
K100_lsystem_perf_ B

Columns in Selected

Columns in View:

a.target_name
a.received_ts
a.request_ts
a.delta_time
a.user_name
a.table_key
a.report_type
a.received_usec
a.dsi_key_id
a.dsi_agg_type
a.ta_sysuptime
a.ta_period
a.ta_samples

Data Type for Column from Key: GAUGE

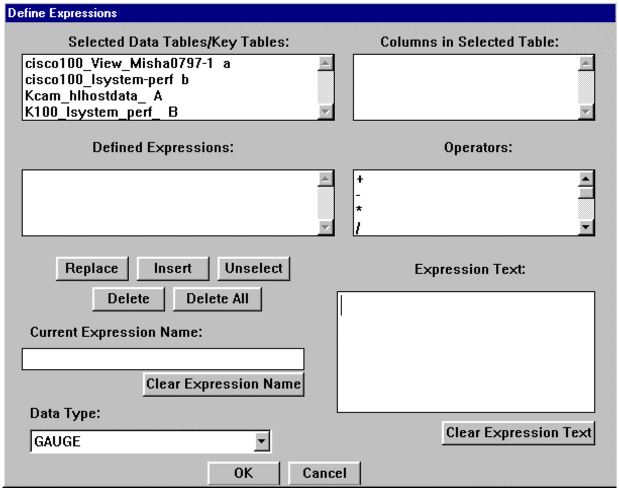
Figure 22-3: Select Columns Window

Access the window by clicking the Select Columns button in the TRENDdbview main window.

It is not necessary to define which columns from the selected data tables will make up the view. However, we recommend that you choose which columns to include. Unless otherwise specified, TRENDdbview creates a view that contains only the default columns headers and footers of the first data table you select. If you want to see data contained in other columns, you need use this dialog box.

Define Expressions Window

The Define Expressions window allows you to determine which (if any) expressions to apply to your view.



Selected Data Tables/Key Tables:

cisco100_View_Misha0797-1 a
cisco100_lsystem-perf b
Kcam_hlhostdata_ A
K100_lsystem_perf_ B

Operators:

+
-
*
/

Data Type: GAUGE

Figure 22-4: Define Expressions Window

It is not necessary for you to define any expressions, and there are no preset default expressions.

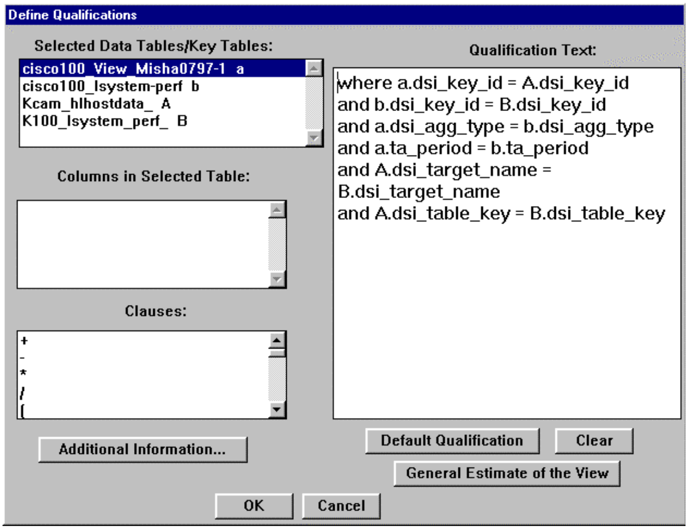
Note: Since TRENDdbview is typically used to view and report on specific data objects, rather than on general information, we recommend that

you apply whatever expressions you consider useful. By doing so, you can achieve better results.

The procedure for defining expressions in a view is described later in this chapter.

Define Qualifications Window

You access the Define Qualifications window through the TRENDdbview main window. In it, you define what qualifications you wish to apply to the view.


Define Qualifications
Selected Data Tables/Key Tables:
Columns in Selected Table:
Clauses:
+
-
*
/
(
Qualification Text:
Additional Information...
Default Qualification
Clear
General Estimate of the View
OK
Cancel

Selected Data Tables/Key Tables:

- cisco100_View_Misha0797-1 a
- cisco100_lsystem-perf b
- Kcam_hlhostdata_ A
- K100_lsystem_perf_ B

Qualification Text:

```
where a.dsi_key_id = A.dsi_key_id
and b.dsi_key_id = B.dsi_key_id
and a.dsi_agg_type = b.dsi_agg_type
and a.ta_period = b.ta_period
and A.dsi_target_name =
B.dsi_target_name
and A.dsi_table_key = B.dsi_table_key
```

Figure 22-5: Define Qualifications Window

It is not necessary for you to qualify your view. Qualifications are at your discretion.

If you wish to use only the default qualifications, you can access the dialog box, then exit without making any changes. The default qualifications will be applied.

If you do not wish the view to use the default qualifications, you can delete some or all of the default settings and click OK. Since TRENDdbview is typically used to view and report on specific data objects, rather than on general information, we recommend that you apply whatever qualifications you consider useful.

Working with TRENDdbview

Starting TRENDdbview

Currently, TRENDdbview is not directly linked or associated with any other Graphical User Interface. For this reason, TRENDdbview must be launched from a command line.

1. Make sure you are logged in as a TREND user.
2. At the command line, type **`trenddbview`** `[RETURN]`. The TRENDdbview main window appears.

From the main window, you can create a new view, or modify an existing one. These commands are located in the File pull-down menu, which also contains the options for saving and running a view.

Building a View

TRENDdbview is used to form a view, combining disparate columns from different data tables, key tables, and views. This creates a virtual data table, and enables you to view and manipulate the specified data. You can then use TRENDbuild to generate TRENDsheet and TRENDgraph reports based on the view. Simply decide which columns of data you wish to include, then use TRENDdbview to create the desired view.

In the course of building a view, you will need to determine its characteristics. This involves up to seven procedures, some of which are optional. Specifically, you need to:

- **Create a new view.**

  Once you have created views, you can open and modify them. Typically, though, you will build a specific view to gather and display data for a specific purpose. Thus, you will most often use TRENDdbview to build new and unique views.

- **Select the tables contributing columns to the view.**

  You will build your view based on columns contained in these tables. In addition, you cannot perform any of the other definitions until you have specified your tables. Previously created views can also act as data tables.

- **Define the qualifications and expressions to be applied to the view.**

  A view does not require either qualifications or expressions. However, if you want your view to run properly without them, you must define their absence. Trying to save or run a view without defining its qualifications will generate a warning message.

- **Select the columns that will make up the view.**

  Since running a view generates a virtual data table, the columns that you choose will be the columns that appear in your view. Choose your columns just as you would when creating a data table.

- **Choose a name for the view.**

  This is done in the TRENDdbview main window, and the name you choose is important, especially when you run the view. This is because of the naming protocol for saving and running views. Opening, closing, and running the view as well as using it as a data table in another view all depend on the name you give your view.

- **Save the view as a file.**

  Your view is saved as an executable file. When you run this file, you generate a virtual table in your database. For this reason, the name and path of the file are significant.

Each of the above tasks can be accomplished by any TREND user. However, before the view can be used to generate reports, it must be created in the TREND database. This is done by "running" the view. Only the trendadm can run the view. For this reason, views are typically defined by the trendadm.

Create a New View

You can start a new view in one of two ways:

- By launching TRENDdbview, if it is not already running. The TRENDdbview main window appears, with "TRENDdbview - untitled" in the Title Bar.

  By default, this is a new view, and you can begin the process of defining its characteristics. Whenever the word "untitled" appears in the Title Bar, the view is a new one.

- By selecting New from the File pull-down menu of the TRENDdbview main window.

  If the name of an existing view appears in the Title Bar, you need to select New from the File pull-down menu in order to create a new view.

Once you have created the new view, you must define its characteristics.

A view is composed of columns taken from existing data tables. You can also include columns from previously created views.

1. You define the view by:

   - Identifying the tables (and/or views) that will make up the view.

- Specifying which columns to include from the selected tables and views.
- Defining any qualifications or expressions you want to apply.

  You must select your data and key tables before making any other definitions.

  After the tables have been selected, you can define qualifications or expressions, or select columns as you wish. There is no set order for these procedures, and you can skip from one to another.

2. Once the parameters are defined to your satisfaction, you need to save the view, then run it.
3. Once you've successfully run a view, you can use TRENDbuild to generate a report file (either *.qss or *.qgr), and view the data in either TRENDsheet or TRENDgraph.

The processes for defining data tables, selecting columns, and assigning expressions and qualifications are detailed in the procedures below.

Define the Data and Key Tables in a View

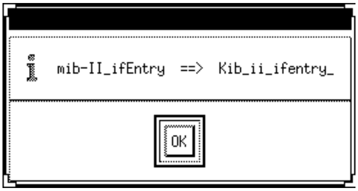
1. Click the Select Data Tables button. The Select data Tables window appears.
2. In the MIB Combo Box, choose a MIB from the list of those available on the current database. Its tables appear in the Available Tables Box.
3. Select one or more tables from the Available Tables Box, and click Add. The selected tables appear in the Selected Data Tables Box, and their associated key tables appear in the Key Tables to Selected Data Tables Box.

Note: At this time, TRENDdbview will only support a maximum of two key tables.

If you highlight a table by mistake, you can deselect it by selecting another table in the Available Tables Box, by selecting a different MIB in the MIB Combo Box, or by clicking Clear.

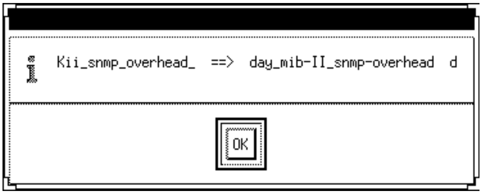
If you add a table by mistake, highlight the table in the Selected Data Tables Box and click Delete. If you decide you don’t want any of the tables you’ve added, click Delete All. The Delete All Button clears the Selected Data Tables Box.

To ascertain which key table references a selected data table, highlight the table in the Selected Data Tables Box, and click the “Data Table => Key Table” Button. An information window pops up, showing the associated key table:



mib-II_ifEntry ==> Kib_ii_ifentry_

To ascertain which data table is referenced by a selected key table, highlight the table in the Key Tables to Selected Data Tables Box, and click the “Key Table => Data Table” Button. An information window pops up, showing the associated key table:



Kii_snmp_overhead_ ==> day_mib-II_snmp-overhead d

4. By default, the first key table that appears in the Key Tables to Selected Data Tables Box also appears in the Key Table for View Box. This is the key table by which the view will be referenced.

   To change the key table used for the view, double-click the other key table in the Key Tables to Selected Data Tables Box. The new key table will appear in the Key Table for View Box in place of the default key table.

5. When the information shown in the Select Data Tables window is correct, click OK. The Select Data Tables window closes, and you return to the TRENDdbview main window.

Once you have specified the tables for your view, you can save and run it. It is not necessary to define either expressions or qualifications. If you choose not to specify which columns to include, your view will include only the thirteen default columns from the primary table (table “a” in the Selected Tables Box of the Select Tables window). If you choose to define any parameters, you can do so in any order. We have chosen to describe the procedure to define qualifications first, but this is merely random. Once you have selected your tables, you can perform any of the remaining procedures in any order you choose.

Define Qualifications for a View

1. Click the Define Qualifications Button. The Define Qualifications window appears, with the default qualifications showing in the Qualification Text Box.

   If you do not wish to apply the default qualifications, click Clear, and the Qualification Text Box is emptied. The Clear Button removes all qualifications.

   If you make modifications to the qualifications Text Box, then decide you wish to restore only the default qualifications, click Default Qualification. The Qualification Text Box will return to its default setting.

   If you add qualification text, then decide you don't want it, you can select and delete it as you would in a typical text editor.

2. To add qualifications, click within the Qualification Text Box. A cursor will appear.

   Any text you add will appear at the site of the cursor, so be sure to position the cursor at the point where you wish to insert your modifications.

   You can add text by typing directly into the Qualification Text Box, by adding text from the Columns in Selected Table Box, or by adding clauses from the Clauses Box.

To add qualification text from the Columns in Selected Table Box, select the appropriate table in the Select Data Tables/Key Tables Box. Its defined columns will appear in the Columns in Selected Table Box. Click the column you wish to insert as qualification text. The text appears in the Qualification Text Box, at the point where the cursor is located.

To add qualification clauses from the Clauses Box, simply scroll to the clause you want, and click it. The clause appears in the Qualification Text Box, at the cursor point.

3. Once you have defined the qualifications you wish to apply to your view, click OK. The Define Qualifications window disappears, and you return to the TRENDdbview main window.

Select Columns for a View

1. Click the Select Column Button. The Columns window appears, with your selected data and key tables in the Selected Data/Key Tables Box. In addition, you will see thirteen (13) default columns in the Columns in View Box.

Note: Do not delete the default columns. They are necessary if the view is to function properly.

2. Highlight one of the tables in the Selected Data/Key Tables Box. Its columns appear in the Columns in Selected Table Box.
3. Select one or more columns from the Columns in Selected Table Box, and click the Insert/Replace Button. The highlighted columns appear in the Columns in View Box.

   You can select more than one column at a time from the Columns in Selected Table Box. There are three ways to do this:

   - You can click-and-drag the mouse pointer over a series of columns. Each column is highlighted.

- You can highlight one column, hold down the SHIFT key, and select another column. Those two columns and all of the columns between them will be highlighted.
- If the columns you wish to select do not appear in sequence, hold down the CONTROL key and select the columns you want. Each of the columns you click will be highlighted.

If you insert columns by accident, you can select and delete them either one at a time or as a group. Highlight the column(s) you wish to delete, and click Delete. Any highlighted column is removed from the Columns in View Box.

If you wish to restore the Columns in View Box to its default setting, click Default. The Columns in View Box is returned to its default condition.

4. When the information that appears in the Columns window is correct, click OK. The Columns window disappears, and you return to the TRENDdbview main window.

Define Expressions for a View

1. Click the Define Expressions Button. The Define Expressions window appears, with your selected data and key tables showing in the Selected Data Tables/Key Tables Box.

   Since it is not necessary to define expressions for any view, TRENDdbview does not apply predefined default expressions. Only those expressions you choose to define will be applied to your view. However, applying expressions is likely to improve your results. For this reason, we recommend that you apply any expression that will help define the view.

2. To define an expression for your view, you need to create the expression text in the Expression Text Box, name the expression, then insert it into the Defined Expressions Box.

   To add an expression, click within the Expression Text Box. A cursor will appear. Any text you add will appear at the site of the cursor.

You can create the expression text by typing it directly into the Expression Text Box, by adding text from the Columns in Selected Table Box, and by adding operators from the Operators Box.

To add expression text from the Columns in Selected Table Box, highlight a table in the Selected Data Table/Key Table Box. Its associated (or defined) columns appear in the Columns in Selected Table Box. Scroll to the column you wish to add, and click it. The text of the column will appear in the Expression Text Box.

To add an available operator to an expression, scroll to the one you want in the Operator Box, and click it. The operator appears in the Expression Text box.

If you add expression text, then decide you do not want it, you can select and delete it as you would in a typical text editor.

3. When the text of your expression is correct, click in the Current Expression Name Text Field. A cursor appears, indicating that you can enter text.
4. Type the name you wish to give to the expression in the Expression text Box, and click Insert. The name of the expression appears in the Defined Expressions Box, and both the Current Expression Name text field and the Expression text Box are cleared.

If you wish to replace or change the name of an expression you have already defined:

- Highlight it in the Defined Expression Box. Its name will appear in the Current Expression Name Text Field, and its text will appear in the Expression Text Box.
- Make the desired changes to either the name, the expression text, data type, or combination of the three, and click Replace. Again, both the Current Expression Name Text Field and the Expression Text Box clear.

If you changed the name, the new name replaces the highlighted name. If you modified the expression text, it replaces the text of the previously defined expression. The previously defined expression is discarded.

If you define an expression, then decide you don't wish to apply it to your view, highlight it in the Defined Expression Box, and click Delete. The expression is removed. If you wish to delete more than one defined expression, you must delete them one at a time. However, if you wish to delete all of your defined expressions, simply click Delete All. This removes all of the defined expressions.

5. Once you have defined the expressions you wish to apply to your view, click OK. The Define Expressions window disappears, and you return to the TRENDdbview main window.

Once you have selected the appropriate tables, and define the qualifications, expressions, and columns you wish, the parameters procedures are complete. It only remains to name, save and run the view.

Name, Save, and Run a View

1. From the View MIB Combo Box, select one of the available MIBs, preferably the MIB of a data table that has contributed at least one column to the view.
2. In the View Name Text Field, type the name you wish to use to identify your view.

   By performing these two steps, you define the name of your view. This name becomes important when you run the view script, which creates the view in your database.

   The name of a view takes the form of MIBname_View_viewname, where MIBname is the MIB you selected from the View MIB Combo Box, and viewname is the text you typed in the View Name Text Field. If, for example, you choose “mib-II” from the View MIB Combo Box, and type “routers1” in the View Name Text Field, then save and run the view, the name of the view appears in mib-ii as “mib-II_view_routers1”.

   ◆ You can use Save As to give the view any file name you wish.

3. From the File pull-down menu, choose Save. The view is saved as an executable file to the default directory.

If you wish to save the view with a different file name, or to a different directory, choose Save As, rather than Save. The Save As Dialog Box appears, allowing you to define in which directory, and under what name the view will be saved.

4. Once you have saved your view, you can run it. By doing so, you create a virtual table in your database. From the File pull-down menu, choose Run The Select File Dialog Box appears, then click Run.

Note: You must save your view before you can run it. The Run command does not interpret the current information in TRENDdbview. Rather, it causes the saved *.vwb file to execute. For this reason, if you do not save your view, it is not available to run.

Once you have run your view, the creation process is complete. The view exists, and acts as a virtual data table. You can use it to generate TRENDsheets and TRENDgraphs, and you can audit it in Data Manager.

Modifying an Existing View

Once you have created a view, you can open and modify it at any time. Then, if you wish, you can save it as a different view, or replace the existing view with the newer version. Replacing an existing view is especially useful if you find that its current parameters are too general.

Open an Existing View

1. Start TRENDdbview, and go to the main window.
2. In the File pull-down menu, select Open The Open File Dialog Box appears.
3. Double-click the view you want to open, or highlight it and click OK. The Open File Dialog Box disappears, the information contained in the view

becomes resident in TRENDdbview, and the view name of the view appears in the TRENDdbview Title Bar.

The view file is now open, and you can make any changes you want to it. However, since the virtual table is created by executing the saved file, your changes will not be reflected until you save and run them.

Save Changes to an Existing View

1. In the File pull-down menu, select Save. The file is updated to reflect the changes you have made.

   Or:

   In the File pull-down menu, select Save As... The Save As Dialog Box appears.

2. In the Save As Dialog Box, define in which directory and under what file name you wish to save the updated view. Click OK.

   > If you wish, you can use the updated view to replace an existing view by highlighting the existing one in this dialog box, then clicking OK.

Once you have saved the updated view, you must run it again to update the virtual table it generates.

Auditing Data and Generating Reports

Once you have created, saved, and run your view, it exists in your database and acts as a data table. You can see the data contained in the view by using TRENDsheet and TRENDgraph. However, before you can use either of these tools to audit the data, you must first build an appropriate report. This is done using TRENDbuild, and the procedure for doing so is described in the chapter on building TREND reports.

A Man Pages

age_files

Allows you to set up the automatic deletion of files from a specified directory.

```
age_files     [-h]
              -i days
              [-l]
              [-p directory]
              [-s]
              [-V]
```

Description

This utility enables you to delete files automatically from the specified directory (and its subdirectories if you also specify the **-s** option).

Options

age_files has the following options:

-h Displays the command line options (help).

-i Specifies the number of days to keep a file. Files whose creation date is older than the number of days specified for this option are deleted.

-l Lists the contents of the target directory but does not delete the files.

-p Identifies the directory to age. The default is the current directory.

-s Removes files meeting the aging criteria specified for the **-i** option from all subdirectories of the directory identified in the **-p** option.

-V Displays the current version and build number of the age_files utility.

db_delete_data

Ages obsolete data out of the database.

```
db_delete_data      [-c #_of_child_processes]
                    [-d debug_level]
                    [-f feeder]
                    [-h]
                    [-i time_index]
                    [-m type]
                    [-s sqlname]
                    [-t tablename]
                    [-u update_statistics]
                    [-U day_of_week]
                    [-V]
```

Description

db_delete_data deletes data from a TREND table when that data has been stored longer than the retention period specified for the table in the TREND database. You can find the data retention period for various tables from the TREND Data Manager display. See "Data Manager" on page 8-1 for details.

db_delete_data is executed automatically according to the schedule specified in the trendtimer.sched file. You can also execute db_delete_data at any time from the command line.

Deleting obsolete data from the database and updating the database statistics page are separate activities. You must invoke db_delete_data with the -u and -U options to update the statistics page.

Options

db_delete_data has the following options:

-c Number of instances of db_delete_data to allow to run concurrently. The default is 1; that is, db_delete_data processes each table one at a time.

-d Sets a debug output level. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out.

-f Deletes data only in tables whose source is the value of this option. Valid values are:

datalyze Legacy tables created/populated by TRENDit (the daily, weekly, and monthly rollup tables that result if the -r option is omitted from the trendit command).

dsi_ee Tables populated by ee_collect.

MW Tables populated by mw_collect.

rmon Tables populated by rmon_collect.

sumit Tables populated by TRENDsum.

tcopy Tables populated by TRENDcopy.

This option is mutually exclusive with the -s and -t options.

If you omit the -f, -s, and -t options, tables from all sources are aged, which is the default.

-h Displays command format help.

-i The number of days that data can be retained in a table. Data that has been in a table for one day more than the number specified is deleted. The default is to use the aging value set for the table in the database. Use Data Manager to view the table’s default aging value. (See “Data Manager” on page 8-1 for details.)

-m Performs key ID-based deletions depending on the value of this option. Valid values are:

data Deletes data using the key IDs selected from the data table.

key Deletes data using the key IDs selected from the property table.

ta_period The data is removed based on time period only.

The default is data.

-s Ages data only in the table specified. Enter the name of the table as shown in the SQL Name column of the Data Manager display. (See “Data Manager” on page 8-1 for details.) This option is mutually exclusive with the -t and -f options. If you omit the -f, -s, and -t options, all data tables are aged, which is the default.

-t Ages data only in the table specified. Enter the name of the table as shown in the Object Name column of the Data Manager display. (See “Data Manager” on page 8-1 for details.)

This option is mutually exclusive with the -f and -s options.

If you omit the -f, -s, and -t options, all data tables are aged, which is the default.

-u Specifies when to update the index statistics page. Valid values are:

1	Before running db_delete_data.
2	After running db_delete_data.
3	Before and after running db_delete_data.

The default is no update statistics. If you specify this option, you must also specify the -U option.

-U Specifies the day of the week to update the database statistics page. Valid values are:

SU	Sunday
MO	Monday
TU	Tuesday
WE	Wednesday
TH	Thursday
FR	Friday
SA	Saturday

There is no default. If you specify this option, you must also specify the -u option.

-V Displays the version stamp for db_delete_data.

Examples

The following command deletes data from each TREND database table one at a time according to the aging criteria specified for each table in the database:

```
db_delete_data
```

The following command:

- Runs concurrent copies of db_delete_data to delete data from all TREND database tables. Five tables are processed concurrently.
- Runs update statistics on Sunday after db_delete_data completes:

```
db_delete_data -c 5 -u 2 -U SU
```

The following command ages the data in the mib-II-ifEntry table, deleting data according to the aging value set for this table in the database:

```
db_delete_data -t mib-II-ifEntry
```

The following command deletes data from the TREND database tables based on time period:

```
db_delete_data -m ta_period
```

Note: To enhance performance in TREND installations using Sybase, make the following change in the %DPIPE_HOME%\lib\trendtimer.sched file. Change

```
24:00+5:00 - - {DPIPE_HOME}/bin/db_delete_data
```

to

```
#24:00+5:00 - - {DPIPE_HOME}/bin/db_delete_data
24:00+24:00+5 - - {DPIPE_HOME}/bin/db_delete_data -c X -u 2 -U Sa
```

where X is the result of the number of system processors multiplied by two child processors. Systems having additional resources could increment the number of child processors in the equation.

The starting time is moved to an earlier time, 5 minutes after midnight, prior to the early morning aggregations of ReportPacks, so that tables are processed, that is reduced in size, and aggregation utilities will perform quicker.

keystats

Performs an update statistics command on each key table in the database.

```
keystats     [-d debug_level]
             [-O sql_timeout]
             [-u]
             [-V]
```

Description

The purpose of this utility is to maintain the index of each key table in order to maximize performance when collectors and data aggregators select against the key tables.

Options

keystats has the following options:

-d Enables you to specify a debug level for debug output. (This option is primarily for development purposes.)

-O Waits for the specified amount of time (in seconds) for activity when a connection is committed to the database. The default is 1800 seconds.

-u Lists the command line options (help).

-V Displays the current version and build number of the utility.

log_backup

Backs up the trend.log file every day to a new file, whose name is based on the day of the week.

```
log_backup
```

Description

An entry to execute this utility is stored automatically in the trendtimer.sched file during the TREND installation procedure. The back-up file names created by this command are:

trend.log.monday
trend.log.tuesday
trend.log.wednesday
trend.log.thursday
trend.log.friday
trend.log.saturday
trend.log.sunday

lpr_launch

Prints TREND reports configured for automatic printing.

```
lpr_launch    [-d debug_level]
              -i interval
              [-V]
```

Description

lpr_launch prints the reports configured for automatic printing by Report Scheduler. lpr_launch is normally invoked by trendtimer once a day to run daily reports, once a week to run weekly reports, and once a month to run monthly reports:

- Daily reports use the midnight-to-midnight period of the previous day as their time period.
- Weekly reports use the most recent Sunday-midnight-to-Sunday-midnight period.
- Monthly reports use the period from midnight on the first day of the month to midnight on the last day of the month.

Reports definitions that are processed by lpr_launch read their data from the database named by the DSQUERY environment variable of the user who invokes lpr_launch. When invoked by trendtimer, lpr_launch uses the DSQUERY value of the user trendadm.

Options

lpr_launch has the following options:

-d	Set a debug output level. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out.
-i	The time interval to indicate whether to run daily, weekly, or monthly reports, specified as follows:

-i 1	Daily
-i 7	Weekly
-i 31	Monthly

This parameter must be specified on the command line. There is no default.

-V	Displays the version stamp for lpr_launch.

Examples

To have lpr_launch run reports configured to be run weekly:

```
lpr_launch -i 7
```

mw_collect

mw_collect is available in two versions: standard and cache. The standard version is a non-cache version that is part of the TREND system. The cache version provides the ability to run without the database and is available for additional purchase.

If you have the cache poller, please see “Instructions for Loading the Cache Version of the Poller” on page A-28.

mw_collect polls nodes for SNMP data.

```
mw_collect    [-a]
              [-A ]
              [-c max_processes]
              [-C wait_time]
              [-d debug_level]
              [-D thread_wait_time]
              [-E clock_error_value]
              [-F min_disk_pct]
              [-g ]
              [-G debug_level_pm]
              [-h hostname]
              -i interval
              [-I check_index]
              [-k]
              [-K suppress_spikes]
              [-L]
              [-M min_filter_value]
              [-n]
              [-N retry_interval]
              [-o timeout]
              [-p max_entries_per_pdu]
              [-P snmp_port]
```

```
[-q log_info_level]
[-r retries]
[-R min_rows]
[-s round_factor]
[-S snmp_version]
[-t tablename]
[-T]
[-u]
[-V]
[-w high_water_mark]
[-W high_water_mark_log]
[-X ]
[-Y delta_time]
[-z child_debug_level]
[-Z debug_log_bcpgateway]
```

Options

mw_collect has the following options:

-a Directs the child collectors of mw_collect to output the collected data in ASCII format. You may want to do this for debugging purposes.

-A Disables archiving of raw data.

The default is archiving. When you use this option, it turns off archiving.

This option is in UPPERCASE.

This option is equivalent to the @bArchive parameter in TRENDpm.

-c Specifies the number of collection processes to run concurrently. When mw_collect starts, it starts child processes that actually do the collections. The default is 5.

-C Specifies the number of minutes that each child process can run. The system kills the child process if it runs longer than the specified number of minutes.

The default is 30 minutes.

This option is in UPPERCASE.

-d Sets the debug output level for the parent instance of mw_collect. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out. You should only use this option for testing in coordination with DeskTalk Systems Technical Support due to the additional overhead it places on mw_collect.

The -z option sets the debug level for the child instances of mw_collect. The -d and -z options can be used together to control the debug level of parent and child mw_collect processes independently.

-D This option specifies how many seconds the parent thread should wait for signals from the collector and bcp_gateway threads.

It is normal to get thread_wait timeout messages in the log file when waiting for signals from the bcp_gateway thread due to potential long running jobs when writing to the database. The program exits when a thread_timeout occurs, unless the thread timeout occurred when all collector jobs are finished and only bcp_gateway threads remained. In this case, the timeout only appears in the trend.log file.

The default is the same value as the -C timeout value, which is 1,800 seconds. For example, if the -C value is 2, which is 2 minutes, then the -D value defaults to 120, which 120 seconds.

This option is in UPPERCASE.

-E Sets the percentage level for valid data. The value is the percentage of difference between the delta values of two Received Timestamps and two System Uptimes. These statistics come from two consecutive raw data samples.

For example, if r1 and s1 are the Received Timestamp and System Uptime for the first sample, and r2 and s2 are the Received Timestamp and System Uptime for the second sample, then the calculation for the value is ((r2 - r1) - (s2 -s1)) * 100 / (r2 - r1). During processing, if the calculated value for the samples exceeds the value set by this option then the samples are rejected.

The default value is 10.

This option is in UPPERCASE.

This option is equivalent to the @zerror parameter in TRENDpm.

-F Specifies the minimum disk space (as percent of total disk space) that must exist on the disk where the $COLLECT_HOME directory resides. If less space exists on the directory, mw_collect does not execute.

The default is 10, which represents 10 percent.

This option is in UPPERCASE.

-g Use 3.5.x compatibility. That is, use the same method for inserting data into the table as version 3.5.x and earlier.

-G Sets the debug output level for the TRENDpm process of mw_collect. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information.

The default is 0, which means no debug output.

Debug output is written to standard out. You should only use this option for testing in coordination with DeskTalk Systems Technical Support due to the additional overhead it places on mw_collect.

This option is in UPPERCASE.

This option is equivalent to the @debug_level parameter in TRENDpm.

-h Names the host to be polled, which must be an SNMP-manageable device. This name will be saved in the database along with the data returned from the node, so in order to be consistent with other data, use the same node name format used when entering nodes in the database.

-i Is the Collection ID, which corresponds to the collection interval in minutes. mw_collect executes the entries in the mw_collection table that have this value in their interval field. How frequently mw_collect is actually run depends on the configuration of TRENDtimer, but the idea is to be consistent so that a collection request with a collection ID of 5 is run every 5 minutes. (See “File Locks” on page A-23 for additional information.)

This option is required.

-I Specifies whether to use existing indices on the upload table or to drop existing indices and then recreate them. The value 1 means that the existing indices on the upload table are used. The value 0 means that the existing indices are dropped and then recreated.

The default is 0.

This option is in UPPERCASE.

This option is equivalent to the @bCheck_index parameter in TRENDpm.

-k Populates the property tables (but not the data tables) for the devices you are polling. Use this option to set up the property tables for devices that you want to poll by key. Then, access the Collect Data Periodically window and set up a polling policy to poll by key for the desired devices. See “Collect Data” on page 7-1 for an explanation of using the Collect Data Periodically window to set up a polling policy.

-K Specifies whether to reject samples if there are spikes. A spike occurs when the value of any counter suddenly goes too high. TRENDpm detects a spike when the value of any counter in the first sample is greater then the value of that counter in the second sample or the difference exceeds the spike threshold. The value of the spike threshold is 2^{31}.

Valid values are:

1 Rejects samples if a spike occurs.

0 Does not reject samples. The default is 0.

This option is in UPPERCASE.

This option is equivalent to the @bSuppress_spike parameter in TRENDpm.

-L Specifies that the collected data be stored locally instead of in the TREND database.

This option is in UPPERCASE.

This option is only available in the cache version of the poller.

-M Sets the minimum filter value. The procedure rejects the sample if the delta value of a counter falls below this value.

The default value is -1, which means to accept the entire sample.

This option is in UPPERCASE.

This option is equivalent to the @line_suppress_value parameter in TRENDpm.

-n Enables distributed polling. If this option is used, mw_collect executes the collection request only if the hostname field in the mw_collection table record for this collection request matches the hostname of the machine on which mw_collect is running. If you omit this option, mw_collect executes all polling requests whose interval matches the value of the -i option, regardless of the hostname specified to do the polling in the polling instructions.

-N Sets the number of seconds the procedure needs to wait in order to acquire a lock on an upload table.

The default value is 10, which is 10 seconds.

This option is in UPPERCASE.

This option is equivalent to the @retry_interval parameter in TRENDpm.

-o Number of seconds mw_collect is to wait for a response after sending an SNMP request. Default is 1 second. (SNMP timeout.)

-p The number of SNMP variables to include in the varbind list in the GET PDU (Protocol Data Unit) request. It is possible to generate a GET request that yields a response that is too long to transmit. The number of variables needed to exceed the maximum response packet size depends mainly on whether the request is being sent over a WAN link, which shortens the allowable packet size compared with a LAN link. The default -p value is 20. For wide-area serial lines, you may need to use a smaller value. If so, begin with a value of 15. If that is successful, increase it by 1 until you begin to receive error messages that PDUs are too large. If 15 is too large, decrease it by 1 until the PDU-too-large messages stop.

-P This option allows the collection of SNMP data from the specified port rather than the default SNMP port of 161.

This option is in UPPERCASE.

-q Sets the log information level for the parent instance of mw_collect. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no log information for the parent instance of mw_collect. The log information is written to the log file called $DPIPE_HOME/tmp/parent_collect_dbg.log. (See Note on page 22.)

-r The number of times mw_collect resends an SNMP request before assuming the target node is not going to respond. The default is 5. (SNMP retries)

-R Sets the minimum number of rows to collect before starting the loading of data into database. The number of rows is an approximation, since there is an assumption that each row is 500 bytes.

The default value is 1000 rows, which means that the loading of the file into the database starts when the size is 500,000 bytes (500 * 1000).

This option is in UPPERCASE.

-s This option rounds off the collection time (ta_period). If the mw_collect parent kicks off a collection at 3:07, and if you are using the default collection option of 300 seconds (5 minutes), the actual ta_period value for the collection will be recorded as 3:05.

-S Specifies the SNMP version. This option is in UPPERCASE.

Valid values for this option are: 1 for SNMP V1
2 for SNMP V2C.

The default uses SNMP V1 packets, which is 1. However, if the table contains a 64-bit counter, the poller automatically switches to SNMP V2C packets, which sets the value to 2.

-t The name of the MIB table that you want to collect. The raw data table must already exist in the database. The tables you can use with this option are shown in Data Manager with type *raw*. Use the value in the Data Manager Object Name column as the argument. (See “Data Manager” on page 8-1 for a discussion of the Data Manager.)

-T Sets fast time conversion. The default is on. When this option is set to off, the timestamp calculations are done using system time functions, which could be very expensive. This option is not available when running in ASCII mode. This option is in UPPERCASE.

-u Displays the command line formats for mw_collect.

-V Displays the version stamp for mw_collect.

This option is in UPPERCASE.

-w Stops collection of data when the database-used size reaches the specified percentage. The default parameter is 90 for 90%.

-X This option to turns off the TRENDpm capability.

This option is in UPPERCASE.

-Y Specifies which clock to use to calculate Delta Time. The value of 1 directs the procedure to use System Uptime to calculate Delta Time. The value of 0 directs the procedure to use the Agent Clock Column for the calculation. The default is 0.

This option is in UPPERCASE.

This option is equivalent to the @bDelta_time parameter in TRENDpm.

-z Sets the debug level for child instances of mw_collect. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out. You should only use this option for testing in coordination with DeskTalk Systems Technical Support due to the additional overhead it places on mw_collect.

The -d option sets the debug level for the parent instance of mw_collect. The -d and -z options can be used together to control the debug level of parent and child mw_collect processes independently.

-Z This option specifies whether to turn on debugging or logging or both for the bcp_gateway process. When the logging option is turned on, the information is written to the log file named $DPIPE_HOME/tmp/bcp_gateway_dbg.log. Valid values for this option are:

1	Turn on logging.
2	Turn on debugging.
3	Turn on both logging and debugging.

This option is in UPPERCASE.

Note: If any logging option is set to on and you are running trendtimer.sched, add an entry to trendtimer.sched to backup log files. To do that add the following command to trendtimer.sched: {DPIPE_HOME}/bin/log_backup -f <input file name>.

Description

mw_collect is TREND's SNMP polling application for generic MIB tables. Its simplest use is to collect a specific table once from a single node and store the information in the TREND database. Fully utilized, mw_collect follows user-defined instructions to poll a dynamic list of nodes anywhere on the network for data at regular intervals and store the results on a local or remote system depending on who requests the information.

The mw_collect command invokes the mw_collect parent poller. The mw_collect parent poller, in turn, invokes multiple children pollers. Each child poller collects a single node/table combination. The -c option specifies the number of child pollers that run concurrently.

When the child poller finishes, the data that is collected is appended to the latest existing holding file for loading into the TREND database. The latest holding file is loaded into the database when the file is at least 500,000 bytes or every child poller has finished. See "Directory Structure" on page A-28 for a discussion of the directories in which these holding files are stored before they are loaded into the TREND database.

Note: Do not use mw_collect to poll for tables in the RMON and token ring RMON MIBs. Do not use mw_collect to poll for interval tables such as the dsx1IntervalTable defined in RFC 1406.

mw_collect calls the following procedures for each table and destination database pair:

- The bcp_gateway procedure updates property and data tables in the TREND database.
- The TRENDpm procedure generates rate data in the TREND database after mw_collect collects all the data and stores it in the database for a particular table.

File Locks

mw_collect allows only one instance of collection for a particular Collection ID to run at the same time. Furthermore, only one bcp_gateway process runs at the same time for a particular Collection ID, table_name, and data database combination.

When the parent poller mw_collect starts, it attempts to place an exclusive lock on the running file in the $COLLECT_HOME/*feeder*/*collection_id* directory. If the lock fails, the poller assumes that the previous instance of the poller is still running and exits. The parent poller releases the lock on the running file when every collector child finishes. Note that the lock releases before the poller finishes loading all the data into database so that the polling continues even though the data loading portion is not finished.

Similarly, when mw_collect starts a bcp_gateway process, it tries to place a lock on $COLLECT_HOME/*feeder*/*collection_id*/*tablename_dbname*.running. If the lock fails, the parent poller assumes that previous job is still running and does not attempt to start the bcp_gateway process. The data is stored locally.

Local Storage of Data

The cache version of mw_collect provides the ability to perform collection without a database. The following table summarizes this feature:

Process	**Failure Point**	**Result Cache Version**	**Result Standard Version**
Parent Poller	Failed to connect at startup.	Use cached information for collection. Store data locally.	Exit
Parent Poller	During initialization, lost connection.	Use cached information (in most cases). Store data locally.	Exit
Parent Poller	Determined during initialization that database is full or transaction log is full.	Store data locally.	Exit

Process	**Failure Point**	**Result Cache Version**	**Result Standard Version**
BCP_gate way	Failed to connect to a given database.	Returns code to calling parent thread. Parent will not attempt to load data into this database during this cycle. Data is stored locally.	Same as Cache version.
BCP_gate way	Lost connection to database, database is full, or transaction log is full.	Removes successfully loaded data from input file, by default; deletes the entire file when operating in ASCII mode (-a). Returns code to calling parent thread. (see above) Depending on when the database connection was lost or the database became full during the bcp_gateway process, the removal of successfully loaded data may not be accurate. This may cause insertion of duplicate data into raw data, which will fail (if there are indexes) and generate error messages in trend.log. bcp_gateway ignores this error message and continues loading new data.	Same as Cache version.
BCP_gate way	Failed loading data for other reason than above.	Removes the entire input file.	Same as Cache version.

Interval Polling

mw_collect is mainly used to poll for SNMP data at a regular intervals. When invoked, mw_collect reads the polling control table in the database for the list of instructions whose intervals match the value of the -i option on the mw_collect command line. Each entry in the list specifies a MIB table to be collected and the device group, specific device, or specific instance from a device to be polled for the table.

This polling policy information is entered automatically when you install a ReportPack. You can also enter or modify polling policy information through the Collect Data Periodically window. Device groups are the View and Type lists defined through the TREND Configure Views and Configure Types windows. See the *TREND ReportPack Guide* for information about using Autopilot. See "Collect Data" on page 7-1 for information about specifying your own polling policies.

mw_collect queries the database for the list of fields in the table to be collected. Then, for each node to be polled, it obtains the node's SNMP Read Community string from the database (as configured in the TREND Configure Nodes application—see "Collect Data" on page 7-1 for details) and polls the node for those fields. Once all target devices have been polled, the returned data is written to the database and mw_collect exits.

Distributed Polling

In the simplest case, all polling instructions, node information, and data tables are on a single system and all polling is done from that system. However, it is possible to maintain polling instructions on one system; store target device names, their community strings, and polling View and Type lists on a second system; execute polling from a third system; store the returned data on a fourth system; and run reports against the returned data from a fifth system. This is the most extensive example of distributing the application, but any combination from the simplest to the most extensive is possible and can be configured according to your requirements.

You can install mw_collect on any system in your network to distribute the polling load to the most efficient locations. The trendtimer program invokes mw_collect and uses trendadm's DSQUERY environment variable value to determine the database to be queried for polling instructions. Since any system can access a central database from anywhere on the network to get its list of polling instructions, you can define the polling policy for an entire network from one central location. Alternatively, you can define polling for each region locally within that region.

Information about nodes to be polled, including their community strings and their membership in polling View and Type lists, is stored in the database called the *Topology Database* (in the TREND Define Users and Groups window). The polling instructions that mw_collect uses include the name of the user who created the request. That user is a member of a polling group, and from that group's definition, mw_collect determines where the Topology Database it needs to execute the current polling instructions is located. mw_collect gets the list of devices to be polled and their community strings from the Topology Database. See "TREND Configuration for trendadm" on page 2-1 for details.

The polling instructions include the name of the system that is intended to do the polling (entered in the hostname field of the Collect Data Periodically window). If mw_collect is run with the -n option, it checks this field to see if the hostname matches the name of the system on which it is running. If they do not match, mw_collect does not execute the poll. Without the -n option, mw_collect ignores the hostname field and runs the poll regardless of any system name specified there.

The group to which the user who defined a polling request belongs also specifies the name of the database to which the collected data is to be written (the *Data* database in the TREND Define Users and Groups window). mw_collect loads the collected data into that database once the polls are completed.

Direct Polling

You can use mw_collect to immediately collect a particular MIB table's values from a particular node on demand by executing mw_collect with the -t option from the command line.

Log File

mw_collect makes log entries in the file $DPIPE_HOME/tmp/trend.log.

Cache Poller

Directory Structure

Temporary files, data files, and cached polling policy files that mw_collect uses are stored under the following directory structure:

`$COLLECT_HOME/`*feeder_name***`/`***collection_id***`/BCP`**

where:

`$COLLECT_HOME` Is the directory that the $COLLECT_HOME environment variable points to (typically, $DPIPE_HOME/collect).

feeder_name Always has the value **MW** for instances of mw_collect.

collection_id Identifies the collection ID, which is a number corresponding to the value of the -i option on the mw_collect command.

BCP Is the directory where data that is ready to be loaded into the database is stored.

Instructions for Loading the Cache Version of the Poller

Perform the following steps to load the cache version of the poller.

1. Install new license key.
2. Rename the standard poller, mw_collect, to another name such as mw_collct_nocache.
3. Rename the cache poller, mw_collect_cache, to mw_collect.

Examples

Interval Polling

To have mw_collect execute all collection requests with a 10 minute interval, invoke mw_collect every 10 minutes with the command line:

```
mw_collect -i 10
```

To have mw_collect execute all collection requests with a 5 minute interval on the machine on which mw_collect is installed, putting a maximum of 20 MIB variables in each packet, trying to poll the node a maximum of 3 times, and waiting 2 seconds between polling attempts, invoke mw_collect with the command line:

```
mw_collect -n -i 5 -p 20 -o 2 -r 3
```

Interval polling entries for mw_collect are in the following file:

```
$DPIPE_HOME/lib/trendtimer.sched
```

Direct Polling

To have mw_collect poll a node called foo.Desktalk.com for the MIB-II ifEntry table use the command:

```
mw_collect -h foo.Desktalk.com -t mib-II_ifEntry -i 59
```

In this example, the -i option is the Collection ID, which means that mw_collect will not read from the database polling policy that corresponds to interval 59.

rmon_collect

Polls RMON probes for RMON MIB data.

Synopsis

Format 1:

```
rmon_collect   [ -B number of minutes <default 60> ] (NT version only)
               [ -c  max number of processes ]
               [ -C number of minutes <default 30> ] (NT version only)
               [ -d  debug level ]
               [ -i  interval ]
               [ -k ]
               [ -n ]
               [ -o  timeout in seconds ]
               [ -p  max entries per pdu ]
               [ -P snmp port number <default 161> ]
               [ -r  number of retries ]
               [ -s round factor(sec) <default 300> ]
               [ -u ]
               [ -V ]
               [ -w high water mark <default 90> ]
```

Format 2:

rmon_collect [**-B** *number of minutes <default 60>*] (NT version only)

[**-c** *max number of processes*]

[**-C** *number of minutes <default 30>*] (NT version only)

[**-d** debug level]

[**-h** hostname]

[**-k**]

[**-n**]

[**-t** tablename]

[**-o** timeout in seconds]

[**-p** max entries per pdu]

[**-P** snmp port number <default 161>]

[**-r** number of retries]

[**-s** round factor(sec) <default 300>]

[**-u**]

[**-V**]

[**-w** high water mark <default 90>]

Description

rmon_collect is TREND's direct SNMP polling application for specific RMON MIB tables. rmon_collect functions in a manner very similar to mw_collect, but has been customized to understand the specialized structures of the various RMON data tables.

Its simplest use is to collect a specific table once from a single probe and store the information in a relational database. Fully utilized, it will follow user-defined instructions to poll a dynamic list of probes anywhere on the network for data at regular intervals and store the results on a local or remote system depending on who requested the information.

Polling Controls

rmon_collect uses the same mechanisms for one time and interval polling as mw_collect. These methods are described in mw_collect(1L). However, rmon_collect should only be used to poll nodes whose type is set to rmon in the TREND Define Nodes window.

The main difference between mw_collect and rmon_collect is that rmon_collect will only poll for the RMON tables which are preloaded into the database by Desktalk Systems. Polling for these tables is setup in the Collect Data Periodically window by selecting them from the Poll For list.

Note: Do not poll for RMON tables by loading the RMON MIBs into MIB-walker. Do not use rmon_collect to poll for interval tables such as the dsx1IntervalTable defined in RFC 1406.

Supported RMON Groups

The RMON groups supported by TREND, and collected by rmon_collect, are:

```
etherStatsEntry
etherHistory
tokenRingMLStatsEntry
tokenRingPStatsEntry
tokenRingMLHistoryEntry
tokenRingPHistoryEntry
hosts
matrix
```

The topN group is not supported because it is not necessary under TREND. Using the capabilities of the relational database, a superset of the topN function is already available.

The alarm, event, filter and capture groups are not supported because they are not applicable to the long term performance monitoring and baselining functions of TREND.

Control Tables

The history, host and matrix groups all use control tables to store information defining the parameters for the various instances of their associated data tables. rmon_collect collects the information in the control table entry each time it collects the data in the data table.

The History Table

rmon_collect looks at each instance of the history table and figures out whether the polling interval -i will cause an overlap in collected data because the polling interval is shorter than the amount of time spanned by the history table instance, or if there will be a gap in the data because the interval is longer than the time spanned by the history table instance (e.g. If a history table instance has 50 buckets, each covering a 30 second interval, polling that instance of the table once an hour with a -i value of 60 will leave a gap of 35 minutes in the data at each poll.). Information about these gaps and intervals is written to $DPIPE_HOME/tmp/trend.log. By understanding how rmon_collect works, and defining history table instances according to the polling interval you use, you can maximize the efficiency of your history table and your polling scheme.

The Matrix Table

rmon_collect only collects the matrixSDEntry version of the matrix table. Using the capabilities of the database, the data in this table can read for any pair of nodes regardless of which is the source and which is the destination node.

Token Ring MAC Layer and Promiscuous Tables

When polling a Token Ring RMON probe, rmon_collect will automatically collect both the MAC Layer and Promiscuous history tables. However, because the user specifies which version of the Token Ring Statistics table they want when they define polling for those tables, only the one selected in the Collect Data Periodically window will be collected by a given polling request.

Log File

rmon_collect makes log entries in the file $DPIPE_HOME/tmp/trend.log.

Options

rmon_collect has the following options:

-B Specifies the number of minutes that each BCP process can run. The system kills the BCP process if it runs longer than the specified number of minutes. The default is 60 minutes. This option is only available on the NT version.

-c The number of concurrent collection processes the master collection mw_collect will start. When mw_collect starts it will start small children processes that actually do the collection, default is 5.

-C Specifies the number of minutes that each collector process can run. The system kills the collector process if it runs longer than the specified number of minutes. The default is 30 minutes. This option is only available on the NT version.

-d Set a debug output level. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out. This option should only be used for testing in coordination with Desktalk Systems Technical Support due to the additional overhead it places on rmon_collect.

-h The name of the host to be polled. This name will be saved in the database along with the data returned from the node, so in order to be consistent with other data, use the same node name format used when entering nodes in the database.

-i The polling interval in minutes. rmon_collect will execute entries in the mw_collection table which have this value in their interval field. Note that how frequently rmon_collect is actually run is determined by the configuration of TRENDtimer, but the idea is to be consistent so that a collection request with an interval of 5 is run every 5 minutes.

-k This option when set will display the key values of the device you are polling. It will only display the interface key not actually do a collection. This is used in the Collect Data-Collect Data Periodically when polling individual node by key for the first time for a device and when the Select Key option is accessed this option is used to get a list of keys.

-n Enable distributed polling. If this option is used, rmon_collect will only execute the collection request if the hostname field in the mw_collection table record for this collection request matches the hostname of the machine on which rmon_collect is running. If you omit this option, rmon_collect will execute all polling requests whose interval matches the value of the -i option, regardless of the hostname specified to do the polling in the polling instructions.

-o The number of seconds rmon_collect will wait for a response after sending a poll. The default is to wait .5 second after the first poll, 1 second after the second and third attempts, and 2 seconds after the fourth before sending a fifth attempt. If this option is used, rmon_collect will wait -o seconds after each poll before retrying, and will try a total of -r times. For nodes that are slow to respond, increasing the timeout value can increase throughput. For example, if a node takes 1.5 seconds to respond, with the default timeout scheme, no response will be seen until after the fourth poll. If the timeout is set to 2 seconds, the first poll will succeed. 2 seconds is a reasonable timeout to set if you are getting a lot of "No response from host" messages from devices that can be polled successfully from the MIBwalker SNMP Tool window.

-p The number of SNMP variables to include in the varbind list in the GET pdu. It is possible to generate a GET request that yields a response too long to transmit. The number of variables needed to exceed the maximum response packet size depends mainly on whether the request is being sent over a WAN link, which shortens the allowable packet size compared to a LAN. The default -p value is 25. For wide area serial lines, you may need to use a smaller value. If so, begin with a value of 15. If that is successful, increase it by 1 until you begin to receive error messages that pdu's are too large. If 15 is too large, decrease it by 1 until the pdu too large messages stop.

-P This allows the collection of SNMP data from another port rather then the default SNMP port of 161.

-r The number of times rmon_collect will send a poll before assuming the target node is not going to respond. The default is 5. If the target node does not respond by the -r poll, rmon_collect will make a "No response from host" entry in $DPIPE_HOME/lib/trend.log and go on to the next target. The time rmon_collect waits between retries is set by the -o option. For devices that are slow to respond, a retry count of 3 combined with a longer timeout value can improve the success rate of your polling.

-s This is used to round off the collection time if started later than the expected time. If the mw_collect parent kicks off a collection at 3:07 and if using the default option of 300sec. (5 min) the actual ta_period value for the collection will be 3:05.

-t The name of the MIB table which you want to collect. The raw data table must already exist in the database. The tables you can use with this option are shown in Data Manager with type “data”. Use the value in the Table Manager “Object Name” column as the argument.

-u Displays the command line formats for rmon_collect.

-V Displays the version stamp for rmon_collect.

-w Stops the collection of data when the database-used size reaches the specified percentage. The collection routine checks the dbstats tables to estimate the current state of the database full size and determine if it should write to the database. The default is 90 for 90%.

Examples

Interval Polling

To have rmon_collect execute all collection requests with a 20 minute interval, invoke rmon_collect every 10 minutes with the command line:

```
rmon_collect -i 20
```

To have rmon_collect execute all collection requests with a 60 minute interval which are to be run from the machine on which it is installed, putting a maximum of 20 MIB variables in each packet, trying to poll the node a maximum of 3 times, waiting 2 seconds between polling attempts, invoke rmon_collect with the command line:

```
rmon_collect -n -i 5 -p 60 -o 2 -r 3
```

Note that the interval polling entries for rmon_collect are in the trendtimer schedule file, $DPIPE_HOME/lib/trendtimer.sched.

Direct Polling

To have rmon_collect poll a probe called rmon1.Desktalk.com for the RMON ethernet History table use the command:

```
rmon_collect -h rmon1.Desktalk.com -t rmon_history
```

TRENDit

Cleans raw data samples and aggregates raw data into periodic summaries.

```
trendit     [-a]
            [-b]
            [-c column_name:value]
            [-d debug_level]
            [-f seconds_past_hour]
            [-h]
            [-m high_ water_mark <default 90> ]
            [-p]
            [-r]
            [-s]
            [-t table]
            [-v host_and_matrix_zero_point_threshold]
            [-V]
            [-w]
            [-z delta_time_error_percentage]
```

Description

TRENDit aggregates raw sample data into hourly, daily and weekly summaries, giving the minimum, maximum, average and total for all counter and gauge fields. It adds a normalized timestamp (ta_period) for each rollup period set to the start of the time period, rounded to the nearest hour, and includes a count of the number of samples (ta_samples) collected during the period.

Options

TRENDit has the following options:

-a To only rollup hourly data that is specified. Values 0 - 23 are valid. If specifying -a 1-13 the rollup process will only generate hourly data from one o’clock in the morning to one o’clock in the afternoon. This only affects the hourly table.

-b Overrides the automatic rejection of data when the delta time is greater than 25 hours.

-c Causes TRENDit to filter out data that is below the specified value for the specified column. TRENDit applies this test after it has computed the delta values for the column. Rows that pass the test are deleted from the rate output table; no rows are deleted from the raw input table.

-d Set a debug output level. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information. The default is 0, which means no debug output. Debug output is written to standard out.

-f The number of seconds after the hour a data collection record can be timestamped and still be considered part of the previous hour. For example, if a group of nodes is to be polled every 20 minutes, data reflecting what happened from 12:00 to 1:00 will be collected at 12:20, 12:40 and 1:00. However, some of the 1:00 samples will come back with a received timestamp of 1:00:01 (or so). To make sure those samples’ data are included in the 12:00 to 1:00 hour, set this option high enough to include the number of seconds after the hour they actually arrived. The default is 60.

-h Display the command line format help.

-m The database-full high-water mark. By default, the rollup process attempts a rollup if the database-full % is less than or equal to 90%. The TRENDit program uses the dbstats table to determine the amount of available space in the database.

-p Causes TRENDit to truncate the RMON host and matrix raw data tables (i.e. delete all rows) after rolling them up. The default is not to truncate the tables.

-r Use this option to only calculate the rates (i.e. intersample delta values). The process stops before running the hourly, daily, weekly, and monthly rollups.

-s This option will handle counter resets that occur on the device agent side. If the device counters are manually reset this option will not allow a high spike in the data which can happen due to the comparison to the sysyuptime which remains constant when counters are reset.

-t The name of the raw data table which you want to roll up. The tables you can use with this option are shown in Data Manager with type "data". Use the value in the Data Manager "SQL Name" column as the argument. The default is to aggregate all tables.

Note that table names that are 30 characters long cause an error message. TRENDit is unable to construct the temporary work table names as <*table-name*>_<*processid*><*date*>. Sybase limits the string to its maximum.

-v Sets the minimum value against which all gauges and counter deltas for each RMON host and matrix table row will be compared to determine whether that row should be rolled up. If the values of ALL these fields in a given row are less than or equal to the -v value, the row is not included in the rollup. Although this can lead to gaps in the rollup data of the RMON host and matrix tables, these gaps are periods when these hosts or pairs of hosts had no activity. By omitting these no-activity rows, considerable space can be conserved in the rollup tables. The default value is 0.

-V Displays the version stamp for TRENDit.

-w Only rollup occurs between a specified time period. If using -w 1-3 the rollup process only rolls up data from Monday to Wednesday; this only affects the daily table. Valid values are: 0 = Sunday, 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday.

-z The percentage difference between the sysuptimes returned with polled data and the local received timestamps of two consecutive raw data samples which TRENDit will allow and still consider the data for the interval valid. If the percentage difference between the changes in these two values is greater than the value set for -z TRENDit ignores the data for this sample interval on the assumption that the difference is due to the polled device being down for part of the interval. (It has been observed that not all system clocks tick at the same speed, sometimes causing the percentage difference to appear artificially high. Set the option high enough to compensate. A value of 30 will usually be enough.) The default value is 10.

Examples

To have TRENDit aggregate the data in all raw data tables in the database, allowing data received up to 3 minutes after the hour to be included as part of the previous hour:

```
trendit -f 180
```

To have TRENDit aggregate the data in the RMON host table, omitting any rows where a host transmitted less than 100 octets, packets and errors between polling samples, and emptying the raw host table after rolling it up:

```
trendit -t rmon_hostdata_ -p -v 100
```

To have TRENDit aggregate the data in the mib-II ifEntry raw data table:

```
trendit -t mib_ii_ifentry_
```

TRENDpm

Manages stored procedures identified by TREND.

```
trendpm         [c code_gen_file]
                [d]
                [db trace_group trace_level]
                [e[a] ]
                [g]
                [h]
                [o[c] output_file]
                [ot target_table_name]
                [pe parameter1=value[,parameter2=value, ...] ]
                [pg parameter1=value[,parameter2=value, ...] ]
                [r]
                [s db_server_name]
                [t table_name, proc_app_type[,proc_class, proc_type] ]
                [v]
```

Options

c Specifies the name of code generated file.

d Delete procedure.

db Set trace parameters. Note that there is no space between the trace group and the trace level.

Valid values for the trace group are:

p	traces the application
d	traces the database
l	traces the library
a	traces all groups

Valid values for the trace level are 1 - 5 where 1 is the lowest verbosity and 5 is the highest verbosity.

e[**a**] Executes the procedure synchronously. Add the **a** option to execute the procedure asynchronously.

g Generates the code.

h or **?** Display the command line format help.

o[**c**] Name of generated or output file. TRENDpm places the header information into the CodeGeneratedFile and overwrites it, if it exists. Adding the c option causes TRENDpm to check for the CodeGeneratedFile; if it exists, TRENDpm exits with an error.

ot Specifies the target table name.

pe Specifies the execution parameters. Do not use any spaces in the parameter list; however, there is a space after the **pe** option.

Valid parameters are:

[**@bArchive** *turn_off_archiving*]
[**@bCheck_index** *check_index*]
[**@bDelta_time** *delta_time*]
[**@bSuppress_spike** *suppress_spikes*]
[**@debug_level** *debug_level_pm*]
[**@line_suppress_value** *min_filter_value*]
[**@retry_interval** *retry_interval*]
[**@zerror** *clock_error_value*]

See "pe Parameters" on page A-45 for the descriptions of the parameters.

`pg`	Specifies the generation parameters. Do not use any spaces in the parameter list; however, there is a space after the **`pg`** option.
`r`	Registers the procedure.
`s`	Specifies the database server name.
`t`	Specifies the table name and the procedure application type, and optionally specifies the procedure class and procedure type.
`v`	Displays the version stamp for TRENDpm.

pe Parameters

`@bArchive`	Enables archiving of raw data. A value of 1 archives the raw data. A value of 0 does not archive the raw data. The default is 1.
`@bCheck_index`	Specifies whether to use existing indices on the upload table or to drop existing indices and then recreate them. The value 1 means that the existing indices on the upload table are used. The value 0 means that the existing indices are dropped and then recreated. The default is 0.
`@bDelta_time`	Specifies which clock to use to calculate Delta Time. The value of 1 directs the procedure to use System Uptime to calculate Delta Time. The value of 0 directs the procedure to use the Agent Clock Column for the calculation. The default is 0.

`@bSuppress_spike`

Specifies whether to reject samples if there are spikes. A spike occurs when the value of any counter suddenly goes too high. TRENDpm detects a spike when the value of any counter in the first sample is greater then the value of that counter in the second sample or the difference exceeds the spike threshold. The value of the spike threshold is 2^{31}.

Valid values are:

1 Rejects samples if a spike occurs.

0 Does not reject samples. The default is 0.

`@debug_level`

Sets the debug output level for the TRENDpm process of mw_collect. Values of 0, 1, 2, or 3 are valid. The higher the number, the more detailed the information.

The default is 0, which means no debug output.

Debug output is written to standard out. You should only use this option for testing in coordination with DeskTalk Systems Technical Support due to the additional overhead it places on mw_collect.

`@line_suppress_value`

Sets the minimum filter value. The procedure rejects the sample if the delta value of a counter falls below this value.

The default value is -1, which means to accept the entire sample.

`@retry_interval`

Sets the number of seconds the procedure needs to wait in order to acquire a lock on an upload table.

The default value is 10, which is 10 seconds.

`@zerror`

Sets the percentage level for valid data. The value is the percentage of difference between the delta values of two Received Timestamps and two System Uptimes. These statistics come from two consecutive raw data samples.

For example, if r1 and s1 are the Received Timestamp and System Uptime for the first sample, and r2 and s2 are the Received Timestamp and System Uptime for the second sample, then the calculation for the value is ((r2 - r1) - (s2 -s1)) * 100 / (r2 - r1). During processing, if the calculated value for the samples exceeds the value set by this option then the samples are rejected.

The default value is 10.

TRENDrank

TRENDrank populates a ranked table on selected metrics from a summary table. It enables you to rank and compare selected metrics from the current period with the previous period.

```
trendrank     [-d debug_level]
              -f definition_file
              [-u]
              [-V]
```

Options

TRENDrank has the following options:

-f Path and name of the definition file.

-d Sets a debug level. Values of 1, 2, and 3 are valid. The higher the number, the more detailed the information. Debug output is written to the standard output destination. This option should only be used for testing in coordination with DeskTalk Systems Technical Support.

-u Displays the syntax for the TRENDrank utility.

-V Displays the version stamp for the TRENDrank utility. Note that this option is in UPPERCASE.

Description

TRENDrank populates an output table that has the following columns for each metric being ranked:

Column	Description
Current value	Current value of the metric. Column name is prefixed with the characters CV.
Current rank	Current rank of the metric among the elements being ranked. Column name is prefixed with the characters CR.
Previous value	Previous value of the metric. Column name is prefixed with the characters PV.
Previous rank	Previous rank of the metric among the elements being ranked. Column name is prefixed with the characters PR.
Delta rank	Delta rank of the metric (absolute value of the change in rank between the current rank position and the previous rank position for the element). Column name is prefixed with the characters DR.
Previous delta rank	Previous delta rank. Column name is prefixed with the characters PDR.

The ranking is typically based on daily level summarized data (although weekly and monthly summaries can also be used as input).

Definition File

TRENDrank needs a definition file to identify the source summary table, the target rank output table, and the columns from the source table that are to be ranked. This definition file has four types of statements; the first three are required:

source_table:*source_summary_table_name* *

destination_table:*target_rank_table_name* *

column:[*target_table_column*=]*source_table_column* [**,** *s*]

sub_variable:*source_table_column*

where *s* has one of the following values:

a	Rank values in the column in ascending sequence.
d	Rank values in the column in descending sequence (default)

* The **source** and **target** statements from the 3.5.x and earlier versions are still available. Please note that they may be discontinued in future releases.

Note: The *table_name* in the source_table or destination_table statement is the alias (group or generic) name, not the name of the item as known to the database; for example, mib_II_ifEntry for the mib_ii_ifentry_table. Similarly, use the alias name for the *source_table_column* in the column or sub_variable statement when that column belongs to a data table. When it belongs to a key table, use the name as known to the database, since there is no group name alias for a key table column.

TRENDrank Sub-Variables

Sub-variables provide a means to restart the rankings within a given date, much the same as a subtotal restarts a summation. With no sub-variables, TRENDrank assigns the rankings according to all the values for the specified metric on a given date. With some sub-variables, TRENDrank assigns the rankings within the same values for the sub-variables. Upon a change in any sub-variable for the date, the ranking begins again.

As an example, suppose there are 10 values for the metric in question on a given date. With no sub-variables, TRENDrank assigns the rankings from 1 to 10. However, if you specify cust_id as a sub-variable, TRENDrank assigns the rankings (beginning at one) to those dsi_key_ids that have the same value for cust_id. When cust_id changes, the rankings begin again at one for the new cust_id.

Each sub-variable must exist as a column in the key table. While similar to the by-variables used by TRENDsum, sub-variables differ in that they do not cause any update of the rank table's key table; nor do they affect the grouping of rows for the rank table. They affect just the contents of the current rank columns in the rank table.

Example

Assume the following statements are stored in a file named **test1.rnk**:

```
source:day_pvc_SDsummary
target:day_pvc_rank
column:in_util=in_utilization,d
column:out_util=out_utilization,d
column:errors,a
```

Enter the following command to execute the TRENDrank utility using **test1.rnk** as input:

```
trendrank -f test1.rnk
```

This command ranks data from a summary table named **day_pvc_SDsummary** and places the ranked columns in an output table named **day_pvc_rank**. The rankings for the output columns associated with **in_utilization** and **out_utilization** are in descending order. The ranking for the output column associated with **errors** is in ascending order. Note that there is a column with the name **errors** in both the source and target tables, so it is not necessary to specify the target alias for the **errors** column.

TRENDstep

TRENDstep conditionally processes each row of a TRENDit input table and stores the result in a new output table. This processing enables you to combine data from multiple rows of an input table into a single row in the output table and compute new columns of data in the output table.

```
trendstep      [-b bcp_packet_size]
               [-d debug_level]
               -f filename
               [-l logsize_high_water_mark]
               [-m dbsize_high_water_mark]
               [-u]
               [-V]
```

Options

TRENDstep options are:

-b Specifies the bulk copy packet size, which can be from 512 to 8192 bytes.

-d Sets the debug output level. Valid values are 1, 2, and 3. The higher the number, the more detailed the information. Debug output is written to the standard output destination. This option is for development purposes.

-f Identifies the input script file. This file contains the commands that generate the TRENDstep output file. Unless the path is specified as part of the file name, this file is assumed to be in $DPIPE_HOME/scripts directory.

-l TRENDstep checks the dbstats tables to determine the current size of the database log and does not run if the log-used size exceeds the percentage specified in this parameter. The default is 90 for 90%.

-m TRENDstep checks the dbstats tables to determine the current size of the database and does not run if the database-used size exceeds the percentage specified in this parameter. The default is 90 for 90%.

-u Displays the syntax of the TRENDstep command.

-V Displays the version of the command.

Description

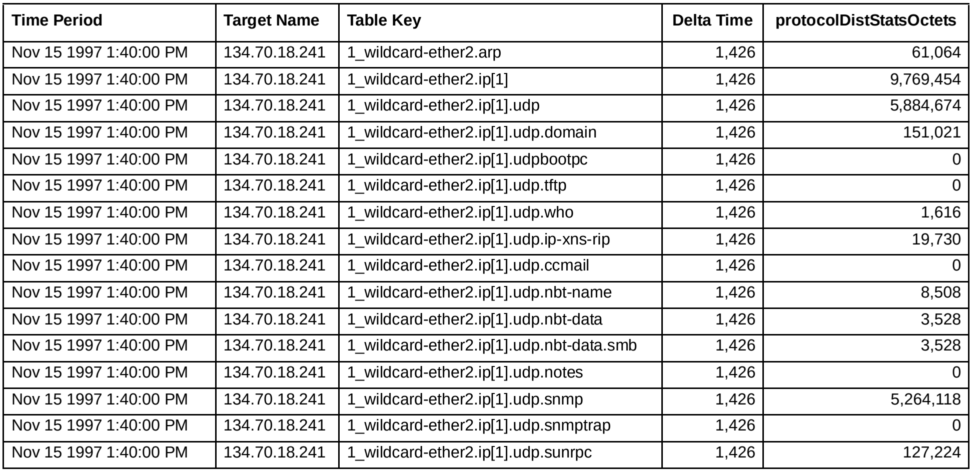
In a typical TREND data table, each row provides many different statistics for a single table key. A report based on this row can show any number of statistics for the table key—but only for that particular table key. Without TRENDstep, if you want to compare the same statistic for many different table keys, you have to generate multiple reports.

TRENDstep, however, allows you to create a new output data table with the statistics you want to see as columns in the table. The power of TRENDstep is its ability to rotate data from multiple rows of an existing data table into a single row in a new data table, while simultaneously aggregating the data into new and useful forms. Among other things, this allows you to compare one segment of data with another in the *same* report, for example, the utilization of Ethernet router interfaces with the utilization of FDDI interfaces, on the same device or across devices.

Example

A good way to illustrate TRENDstep is to see how it converts an RMON2 data table. Figure A-1 shows part of an RMON2 data table with many rows of data related to network traffic. The columns in the table are Time Period, Target Name (device identifier), Table Key, Delta Time, and Octets. Each row of the table represents the octet count for a specific communications protocol, as recorded by an RMON2 agent on device 134.70.18.241, which is identified in the Target Name column.

Note that the Time Period is the same (1:40 p.m.) for each row. The figure shows all of the tracked RMON2 protocols for a single polling period (ta_period), which is NOV 15 1997 1:40:00 PM. For the purposes of this example, we focus on the data for this polling period.



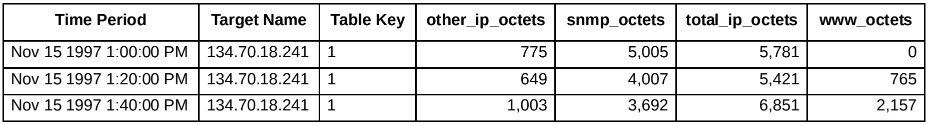
Time Period	Target Name	Table Key	Delta Time	protocolDistStatsOctets
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.arp	1,426	61,064
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1]	1,426	9,769,454
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp	1,426	5,884,674
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.domain	1,426	151,021
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udpbootpc	1,426	0
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.tftp	1,426	0
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.who	1,426	1,616
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.ip-xns-rip	1,426	19,730
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.ccmail	1,426	0
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.nbt-name	1,426	8,508
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.nbt-data	1,426	3,528
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.nbt-data.smb	1,426	3,528
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.notes	1,426	0
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.snmp	1,426	5,264,118
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.snmptrap	1,426	0
Nov 15 1997 1:40:00 PM	134.70.18.241	1_wildcard-ether2.ip[1].udp.sunrpc	1,426	127,224

Figure A-1: Example RMON2 Data Table

If you want to know how many octets are recorded for a particular protocol, this table suits your purpose. Each row represents a tracked protocol. Using TRENDsheet or TRENDgraph, you can generate a report to show this count for any single protocol.

However, if you want to generate a single report that shows total IP octets, total WWW octets, total SNMP octets, and total IP octets that are not either WWW or SNMP, you need to use TRENDstep to create a different table with each of these statistics as a column in the table.

That is, as shown in Figure A-1, the TRENDstep input table has this information in separate rows. The TRENDstep output table, shown in Figure A-2, has this information in different columns of the same row:



Time Period	Target Name	Table Key	other_ip_octets	snmp_octets	total_ip_octets	www_octets
Nov 15 1997 1:00:00 PM	134.70.18.241	1	775	5,005	5,781	0
Nov 15 1997 1:20:00 PM	134.70.18.241	1	649	4,007	5,421	765
Nov 15 1997 1:40:00 PM	134.70.18.241	1	1,003	3,692	6,851	2,157

Figure A-2: Example TRENDstep Output Table

The Output Table shows three Time Periods: 1:00, 1:20, and 1:40. We are only interested in the third row for the 1:40 Time Period. The output table has seven columns:

Time Period The polling period of the data.

Target Name The device (RMON2 agent) that reports the data.

Table Key The Table Key for the row. In this example, this value represents the interface number of an RMON2 probe. It is a new Table Key, created by TRENDstep.

other_ip_octets The total number of octets counted that were not either WWW or SNMP traffic.

snmp_octets The total number of SNMP octets counted.

total_ip_octets The total number of octets counted including SNMP, WWW, and other traffic.

www_octets The total number of WWW octets counted.

The value shown for snmp_octets in Figure A-2 is 3,692, which differs from the value shown in Figure A-1 for SNMP octets, which is 5,264,118. These values differ because TRENDstep has converted the value in Figure A-1, which is a simple counter, to a rate value in Figure A-2. The Delta Time for that row in Figure A-1 is 1,426 seconds. Thus, if the RMON2 agent recorded 5,264,118 octets of SNMP traffic for the polling period, SNMP octets were passed at a rate of 3,692 octets per second (the value shown in Figure A-2). All statistics in the output table have been converted to octets per second.

Now look at the column total_ip_octets; for the 1:40 p.m. polling period, the value in this column shows that the RMON2 agent for device 134.70.18.241 recorded an average of 6,851 octets per second of IP traffic. We can see that of the total traffic:

- SNMP traffic averaged 3,692 octets per second
- WWW traffic averaged 2,157 octets per second
- Combined, all other IP traffic averaged 1,003 octets per second

Most importantly, all of these statistics are now in a single row of a data table, which allows you to generate a graphical report that includes all of these statistics, as shown in Figure A-3.

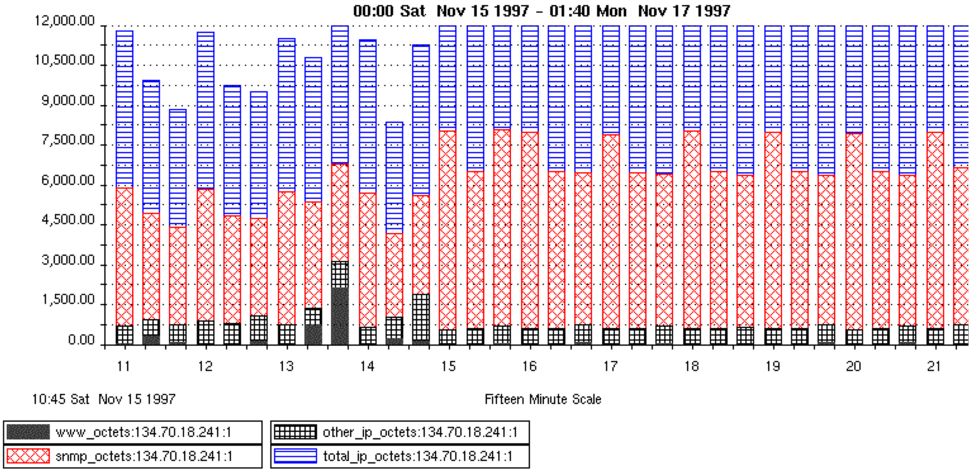


Figure A-3: Example Stacked Bar Graph, Based on Example Output Table

Figure A-3 shows a part of the bar chart generated from the example output table. The actual report shows the specified statistics over a 24 hour period. Since we are only interested in the 1:40 p.m. time period, Figure A-3 shows only that part of the report that includes 1:40 p.m. The line, or bar for 1:40 p.m. is a representation of the IP traffic recorded by RMON2 agent on device 134.70.18.241 for that time period. The stacked bar chart format is useful in that it provides a visual basis of comparison. For precise, numeric values, the report shown in Figure A-2 with TRENDsheet is more suitable.

TRENDstep Input File

A TRENDstep input file, by convention, has the file extension **.rot**, although you can use any extension. You can use the following statements in this file:

Statement	**Description**
source_table:*table_name***;** *	Name of the input table. The table name you enter on this line may be the SQL (real) Table Name or the Object (Alias) Table Name. Also, TRENDstep cannot run against raw data. You should only specify a data table against which TRENDit or TRENDsum (or both) has been run. This statement appears as the first executable line of the input file and is required.

Statement	**Description**
destination_table:*table_name***;** [*]	Name of the output table. The table name you enter on this line must be the Object (Alias) Table Name for a raw data table. TRENDstep automatically adds the appropriate prefix to reflect the table's rollup level (rate, hourly, daily, etc.). For example, if you specify mytest for this name and the input table contains daily rollups, TRENDstep creates an output table with the SQL name of Dytest and an Object name of day_mytest. This statement appears as the second executable line of the input file and is required. TRENDstep creates this table if it does not exist; or TRENDstep appends data to this table if it already exists.
#define IGNORE_NULL;	Specifies to ignore default rules for NULL values. When a null value appears in an expression, the default result is NULL. String variables are exceptions: A NULL string value is assumed to be an empty string. If this statement is used, a NULL value in + and - expressions is treated as zero. For more information, see "Handling Null Values in Expressions" on page A-65.

Statement	Description
#define TIMEADJUST;	This statement enables the ta_period/ delta_time adjustment (zero rate row assumption) mechanism for rate tables. That is, TRENDstep determines the minimum polling cycle for the entire table, and then adjusts corresponding ta_period and delta_time columns to account for the resulting polling cycle. When you use this statement, make sure the polling interval for all rows in the input table is the same; otherwise, you will generate invalid data. If this statement is omitted and the input table has rate data, the output column will have rate values for counter columns. That is, we will divide counters by delta_time (this is the default behavior), and define this column to be of gauge data type.
#define IGNORE_TIMEADJUST;	This statement disables the ta_period/ delta_time adjustment mechanism, which means that TRENDstep ignores any differences that may occur in the delta_time column. This statement is valid only if the input table is a rate table. See the description for **#define TIMEADJUST** for the default behavior.

#FINAL;
#ENDFINAL;
	These statements enclose the statements for the final block. The statements in the final block are executed only once on the result columns. Only output data columns and output key columns can appear inside a final block.

Statement	Description
assignment	There are two types of assignment statements: a = b, which means a equals b. a =+ b, which means a equals a + b. To initialize a column to 0, use the following statement: *column_name* `= +0;`

by_variable:*expression***;** *	Specifies the dsi_target_name and dsi_table_key values for the output key table. The first by_variable identifies the target name; all the other by_variables define the table key. This statement is required. Use the expression **KEY.***column_name* to identify a key table column name.
	Examples:
	`by_variable:KEY.dsi_target_name;`
	`by_variable:KEY.dsi_data_source;`
	`by_variable:dsi_agg_type;`

case (*logical_operator expression)***;**	This statement defines a case within the switch block. Subsequent statements in the case are applied to all rows from the input file that match the specified expression. The next **case**, **default**, or **endswitch** statement signals the end of the case.
column_name=*expression***;**	This statement defines a column in the output table. The order of the column statements is the order of the columns in the output table, if created. This statement is required.

Statement	**Description**
comment	Comments begin with the characters /* and end with the character */. Examples: /*this is a comment.*/ /*this is a comment.*/ type:ifdescr[128]; /*comment can be on the same line as another statement too*/
default;	This statement defines the default case within the switch block. Subsequent statements in the default case are applied to all rows from the input file that are not processed by any preceding case in the switch block.

if (*expression***);**
elseif (*expression***);**
else;
endif;
	These statements define the processing in an if block. One or more column=expression; statements can follow the **if** and **elseif** statements.

switch (*expression***);**
endswitch;
	These statements enclose the statements for the switch block. The statements in the switch block are executed for each row of the input table that satisfies the expression given on the **switch** statement.

type:*column_name***[***size***];**	The type statement defines the length of a column that contains string data. The default size is 20 bytes; the maximum size is 255 bytes. The [ ] characters enclose the size, and are required. Example: type:ifdescr[128];

* The **byvariable**, **input_table**, and **output_table** statements from the 3.5.x and earlier versions are still available. Please note that they may be discontinued in future releases, since TREND has standardized on the new labels for all rollup programs.

The following statements are required in the input file:

- **source_table:***table_name***;**
- **destination_table:***table_name***;**
- *column_name*=*expression***;**
- **by_variable:***expression***;**

Every command in an input file must end with a semicolon (;). Blank spaces in the input file are ignored, unless they are part of a character string.

These statements and their components are described in detail below.

The length of the *table_name* in the **destination_table** statement should have a maximum of 31 characters; otherwise, if the name is longer TRENDstep creates an invalid key table.

If you have inadvertently specified a table name that is greater than 31 characters in the destination_table statement and already executed TRENDstep to create the invalid key table, perform the following steps to remove it:

1. Kill the TRENDstep process after it has created the base table and invalid key table.
2. Use the ISQL utility to delete the invalid key table name from dsi_key_tables, as follows:

   ```
   isql -Udsi_dpipe -Pdsi_dpipe
   1> delete from dsi_key_tables
   2> where dsi_key_table = “invalid-key-tablename”
   3> go
   ```

3. Use the ISQL utility to delete the invalid key table:

   ```
   1> drop table invalid-key-tablename
   2> go
   ```

Column Definition

You must use a column statement to define each column in the output table. The column statement has the form:

column_name=*expression***;**

where *expression* can contain the following components:

- Other column names.
- Any rational number (does not have to be an integer), for example, 5, 12.5.
- Strings. Strings consist of any sequence of characters enclosed in single quotes. Examples:

  ‘this is a string.’

  ‘ ‘ is a blank string (single quotes enclose a space character).
- Mathematical operators. See “Mathematical Operators” on page A-64.
- Logical operators. See “Logical Operators” on page A-64.

Using Column Names in Expressions

Any string that is not a keyword, constant number, literal, or comment is considered to be a column name. There are four ways you can reference a column name, depending on the column type and how it is used in an expression:

Column Type	How to Reference
input data	Names a column in the input data table. Use the column SQL name.
input key	Names a column in the input key table. Use the **KEY.***column_name* syntax.

Column Type	**How to Reference**
output data	Names a column in the output data table. On the left side of an assignment expression, use just the column SQL name; otherwise, use the **ROTATE.***column_name* syntax.
output key	Names a key column in the output key table. On the left side of an assignment expression, use the **KEY.***column_name* syntax. On the right side of an assignment expression, use the **RKEY.***column_name* syntax.

Mathematical Operators

The following mathematical operators are defined:

+, -, *, /

Expressions are evaluated from left to right. All mathematical operators are equal; use parentheses to ensure proper expression evaluation order.

Logical Operators

The following logical operators are defined:

Operator	**Meaning**
=	Equal To
!=	Not Equal To
<	Less Than
>	Greater Than
<=	Less Than or Equal To

Operator	Meaning
>=	Greater Than or Equal To
&&	Boolean AND
||	Boolean OR

Handling Null Values in Expressions

The following table summarizes the rules for handling null values during arithmetic operations:

Expression	Default Result	Result if #define IGNORE_NULL Statement is Used
NULL + a	NULL	a
NULL – a or (a – NULL)	NULL	-a (a)
NULL / a or a / NULL	NULL	NULL
NULL * a	NULL	NULL

The following table summarizes the rules for handling null values during logical operations:

Expression	Result
NULL = NULL	TRUE
NULL = not null value	FALSE
NULL != NULL	FALSE

Expression	**Result**
NULL != not null value	TRUE
NULL < (>) NULL	FALSE
NULL < (>)not null value	FALSE
Not null value < (>)NULL	FALSE
NULL <= (>=) NULL	TRUE
NULL <= (>=)not null value	FALSE
Not null value < =(>=)NULL	FALSE
NULL AND (OR) anything	FALSE

Final Block

The statements in a final block are executed only once on the result columns. Only output data columns and output key columns can appear inside a final block.

Figure A-4 shows an example of a Final Block:

```
#FINAL;
if ((RKEY.speed = NULL) || (ROTATE.ipoctets = NULL) ||
        (RKEY.speed = 0));
                iputilization = 0;
else;
        iputilization = (ROTATE.ipoctets * 8)*100 / RKEY.speed;
endif;
if (ROTATE.iputilization < 5);
        grade = ‘BAD’;
else;
        grade = ‘GOOD’;
endif;
#ENDFINAL;
```

Figure A-4: Example of a Final Block

Switch Blocks

Figure A-5 shows the construction of a switch block:

```
switch (expressions);
    case (<logical operator> expression);
       column_name = expression;
       column_name =+ expression;
    case (<logical operator> expression);
       column_name = expression;
    default;
       column_name = expression;
endswitch;
```

Figure A-5: Example of a Switch Block

Rules for the expressions used in a switch block include:

- Expressions can contain constants, column names, +, -, /, *, (, and).
- Expressions are evaluated from left to right; no operator precedence is observed. However, expressions within parentheses are evaluated first.
- The following logical operations are allowed: =, != , <, <=, >, >=.
- The && and || boolean operators are allowed in a switch block, but might make it difficult to understand the logic. Use IF/ELSE constructions instead if boolean logic is needed.
- If one of the case statements inside a switch block is true, the rest are skipped.
- A switch block's default case is executed only if every other case inside the switch block is false.

IF Blocks

Figure A-6 shows the IF block construction:

```
if (expression);
    column=expression;
    ...
else if  (expression);
    column=expression;
    ....
else if (expression);
    column = expression;
    ...........
else;
    column = expression;
endif;
```

Figure A-6: IF Block Construction

String Comparison

To perform wildcard character comparison use the percent (**%**) symbol to represent any sequence of one or more characters inside a string. Only equal and not equal comparisons are allowed if wildcard characters are used.

TRENDstep Output Table Creation

If the specified output table does not exist, TRENDstep creates the new data table and its associated key table.

The new data table contains the same header and footer columns that appear in the input_table plus all the data columns that appear on the left hand side of expressions (i.e., *column_name* = *expression***;**). The columns are added in the order of the column statements.

The new key table contains the standard key table columns plus all of the KEY columns that appear on the left hand side of expressions (i.e., **KEY.***column_name* = *expression***;**).

TRENDstep tries to determine the type of the output column from the type of the expression. If constants are used as part of the expression, a floating point number indicates that the resulting column is of counter type; an integer number indicates that the resulting column is of gauge type.

Examples of Input Script Files

Three examples of TRENDstep input script files follow. You may see additional examples by reviewing the files with the .rot extension that are available with the installed ReportPacks in the $DPIPE_HOME/packages directory.

Example1

Figure A-7 is an example of an input script file. It is used to generate the output data table in Figure A-2, “Example TRENDstep Output Table,” on page 55.

```
source_table:Rmon2_statsdata_;
destination_table:rmon2_protdist;
#define IGNORE_NULL;
KEY.if_speed=KEY.ifspeed;
if ((KEY.nt_layer = ’ip[1]’) && (KEY.ap_layer = ’www-http’));
    www_octets = +diststatsoctets010;
    other_ip_octets = +(0.0-diststatsoctets010);
else if ((KEY.nt_layer = ’ip[1]’) && (KEY.ap_layer = ’snmp’));
    snmp_octets = +diststatsoctets010;
    other_ip_octets = +(0.0-diststatsoctets010);
endif;
switch (KEY.dsi_table_key);
    case (=’%.ip[1]’);/* count network layer ip octets */
        other_ip_octets = +diststatsoctets010;
endswitch;
#FINAL;
    total_ip_octets = ROTATE.other_ip_octets +
    ROTATE.www_octets + ROTATE.snmp_octets;
#ENDFINAL;
by_variable:KEY.dsi_target_name;
by_variable:KEY.dsi_data_source;
```

Figure A-7: Example of a TRENDstep Input Script File

Figure A-8 shows the same Input Script File with comments on the significance of each line or block.

Line Item	**Comment**
source_table:Rmon2_statsdata_;	Identifies the source table.
destination_table:rmon2_protdist;	Identifies the output table.
#define IGNORE_NULL;	Specifies how TRENDstep will deal with null values. See "Handling Null Values in Expressions" on page A-65.
KEY.if_speed=KEY.ifSpeed;	Include the column if_speed in the resulting key table, and get its value from the ifSpeed column of the source key table.
if ((KEY.nt_layer = 'ip[1]') && (KEY.ap_layer = 'www-http'));	If a row in a source data table contains statistics about WWW traffic over IP, then ...
www_octets = +diststatsoctets010;	Include the statistics in the resulting WWW octet count column of the destination table.
other_ip_octets = +(0.0-diststatsoctets010);	Avoid counting this data twice, by subtracting this count from the count of all other IP counts.
else if ((KEY.nt_layer = 'ip[1]') && (KEY.ap_layer = 'snmp'));	If a row in a source data table contains statistics about SNMP traffic over IP, then ...
snmp_octets = +diststatsoctets010;	Include the statistics in the resulting SNMP octet count column of the destination table.
other_ip_octets = +(0.0-diststatsoctets010);	Avoid counting this data twice, by subtracting this count from the count of all other IP counts.

Line Item	**Comment**
endif;	This statement ends the IF block.

switch (KEY.dsi_table_key);
case (=’%.ip[1]’);
other_ip_octets = +diststatsoctets010;
endswitch;
	If a row in the source data table contains statistics about IP only, include the octet count in the network layer ip octet count of the output table. This column represents all of the counts of IP traffic other than WWW and SNMP, which is the reason these counts were subtracted from the other_ip_octets field in the preceding IF blocks.

#FINAL;
total_ip_octets = ROTATE.other_ip_octets +
ROTATE.www_octets +
ROTATE.snmp_octets;
#ENDFINAL;
	This specifies the creation of an additional column in the output table, which contains the total count of all IP traffic (i.e. WWW counts + SNMP counts + other counts).

by_variable:KEY.dsi_target_name;
by_variable:KEY.dsi_data_source;
	This specifies that the output table will contain one row for each target name/interface combination (for each ta_period in the input file).

Figure A-8: Example Input Script File with Comments

Example 2

```
/* this file produces a break down of IP and Other packets */
/* in addition it grades the usage, if the value of the Other packets column is >
0.04 % of ippkts */
/* then the grade = bad */

#define TIMEADJUST;
source_table:Rmon2_statsdata_;
destination_table:rmon2_protocols;

switch (KEY.dsi_table_key);
        case (=’%.ip’);    /* nt layer = ip */
                ippkts =+ oldiststatspkts009;
        case (= ’%.%.%’);  /* has layer above nt layer */
                /* ignore it */
                * skip just physical layer */
        case (!=’%.ip’) /*nt layer without IP, must be other */;
                otherpkts =+ oldiststatspkts009;
endswitch;

/*The Switch Block below should properly be in a Final Block, as it should only be
executed once on the resulting rows---It is shown as it is for example purposes
only.*/
switch (ROTATE.otherpkts / ROTATE.ippkts * 100.0);
        case (<= 0.04);
                grade = ’GOOD’;
        default;
                grade = ’BAD’;
endswitch;

by_variable: target_name001;
by_variable: KEY.dsi_data_source;
```

Example 3

```
/* this example does something very similar to the one above, but here we use if
statements */
/* this report will produce separate columns for snmp packets, snmp trap
packets, www packets, other ip packets, and total ip pkts */

#define TIMEADJUST;
source_table:Rmon2_statsdata_;
destination_table:rmon2_protocols;


switch (KEY.ap_layer);
        case (=’snmp’);
                snmpcount=+ oldiststatspkts009;
                otheripcount=+ (0.0-oldiststatspkts009);
        case (=’snmptrap’);
                snmptrap=+oldiststatspkts009;
                otheripcount =+ (0.0-oldiststatspkts009);
        case (=’http’);
                wwwcount=+oldiststatspkts009;
                otheripcount =+ (0.0-oldiststatspkts009);
        case (=’www-http’);
                wwwcount=+oldiststatspkts009;
                otheripcount =+ (0.0-oldiststatspkts009);
endswitch;

if ((KEY.nt_layer = ’ip’) && (KEY.tr_layer = ’ ’));
        otheripcount=+ oldiststatspkts009;
endif;

#FINAL;
        totalip = ROTATE.otheripcount + ROTATE.wwwcount +
        ROTATE.snmpcount +ROTATE.snmptrap;
#ENDFINAL;

by_variable: target_name001;
by_variable: KEY.dsi_data_source;
```

TRENDsum

Calculates various statistics such as averages, percentiles, baseline values, and forecast values.

```
trend_sum     [-a  baseline table name]
              -b  by_variable1 [ ... by_variablen]
              -c  column_spec1 [... column_specn]
              [-e  destination_table ]
              [-f  file]
              [-g]
              [-h]
              [-m ]
              [-n  null_percentage]
              [-r]
              -t  source_table
              [-V]
              [-w  first_day_of_week]
              [-y  baseline_days]
```

Options

TRENDsum has the following command line options:

-a Optional name of the destination table for TRENDsum baseline output. The full name of the destination table will be SD*nn* plus this name, where *nn* is the number of days used for the value of the **-y** option.

You may not use both the **-a** and **-e** options.

-b The specification for the grouping by-variable (see “TRENDsum By-Variables” on page A-81), which has the format:

[**alias** =]*column* or *keyword*

where:

column	Is the name of a column in the source table.
keyword	Has one of the following values:
keyid	Group by the column named dsi_key_id_
ta_period	Group by ta_period
hour	Group by hour
day	Group by day
week	Group by week
month	Group by month
quarter	Group by quarter
year	Group by year
day_of_week	Group by day of week for baseline table.
day_of_week_by_hour	Group by day of week by hour for baseline table.
hour_of_day	Group by hour of day for baseline table.

-c The specification for a column in the output table, where column_spec has the format:

[alias =] column_expr:*statistic1* [,...*statisticn*], where *statistic* is at least one of the following:

avg	Average
cnt	Count of samples
dtt [*value*]	Number of days until *value* is reached, where *value* is the threshold value the user specifies
f30	Projected value in 30 days
f60	Projected value in 60 days
f90	Projected value in 90 days
mad	Delta time when the maximum value occurred
mat	Time when the maximum value occurred in the source table
max	Maximum value
med	Median
mid	Delta time when the minimum value occurred
min	Minimum value
mit	Time when the minimum value occurred
per90	90th percentile
per95	95th percentile
per98	98th percentile

rct [*min* **-** *max*] Count of samples greater than or equal to the *min* value and less than the *max* value (i.e., count of samples >= min and < max)

Note: The definition of this statistic has changed. See the -g option to use the version 3.5.1 and earlier definition.

std Standard deviation (the square root of the variance)

tct [*value*] Count of samples greater than or equal to *value* (i.e., count of samples >= value)

Note: The definition of this statistic has changed. See the -g option to use the version 3.5.1 and earlier definition.

tot Total

vct [*value*] Count of samples equaling *value*

wav Weighted average

-e Name of the destination table for the TRENDsum output. This option is required for a non-baseline table.

For a baseline table, the full name of the destination table will be SD*nn*, where *nn* is the number of days used for the value of the **-y** option, plus the source table name (**-t**) plus this name.

You may not use both the **-a** and **-e** options.

-f Name of an input file containing the aggregation instructions.

-g Use version 3.5.1 compatibility for the **rct** and **tct** statistics in the -c option.

rct [*min* **-** *max*] Count of samples between the *min* and *max* values (i.e., count of samples > min and < max)

tct [*value*] Count of samples above *value* (i.e., count of samples > value)

-h Display the syntax for this utility.

-m Use 3.4.x compatibility. That is, apply 3.4.x TRENDsum summarization algorithms. See “Changes in TRENDsum 3.5.1 from 3.4.x” on page A-86.

-n Percentage of samples that can have a null value. Default is 50. For example, assume -n 10 is specified for this parameter. If there are 100 samples and 11 are null (due to collection agent problems, faulty collections, or the device being unavailable at the time the sample was taken), the value for the statistic is null.

-r Replace an existing destination table with the same name.

-t Name of the table that is providing the input to TRENDsum.

-V Display the version of this utility.

-w First day of the week, specified as: Mon, Tue, Wed, Thu, Fri, Sat, or Sun. Default is Mon. This parameter is not case-sensitive.

-y Number of days in the baseline period.

Description

TRENDsum Formulas

Formulas for computing the statistics are:

tot $\Sigma\ val$

where val is the resolved value of the expression to which the tot function is being applied.

avg tot / cnt

where cnt is the number of samples being summarized.

wav $\Sigma\ (val * \Delta t) / (\Sigma\ \Delta t)$

std $sqrt\ ((cnt * \Sigma\ (val * val) - tot * tot) / (cnt * (cnt-1)))$

med vr[floor(.50 * (cnt-1))]

where vr is an array of vals sorted into ascending order.

p90 vr[floor(.90 * (cnt-1))]

where vr is an array of vals sorted into ascending order.

p95 vr[floor(.95 * (cnt-1))]

where vr is an array of vals sorted into ascending order.

p98 vr[floor(.98 * (cnt-1))]

where vr is an array of vals sorted into ascending order.

d $cnt * \Sigma\ (\Delta t * \Delta t) - \Sigma\ \Delta t * \Sigma\ \Delta t$

where Δt = ta_period column value of this row minus the ta_period column value of the first row in the group.

b $(cnt * \Sigma\ (val * \Delta t) - tot * \Sigma\ \Delta t) / d$

xbar $\Sigma\ \Delta t / cnt$

ybar tot / cnt

a ybar - b * xbar

f30 a + b * (Δtg + 86400 * 30)

where Δtg is the ta_period column value of the last row in the group minus the ta_period column value of the first row in the group.

f60 a + b * (Δtg + 86400 * 60)

where Δtg is the ta_period column value of the last row in the group minus the ta_period column value of the first row in the group.

f90 a + b * (Δtg + 86400 * 90)

where Δtg is the ta_period column value of the last row in the group minus the ta_period column value of the first row in the group.

dtt ceil(((threshold_value - a) / b - Δtg) / 86400)

where threshold_value is the user-specified threshold value and Δtg is the ta_period column value of the last row in the group minus the ta_period column value of the first row in the group.

TRENDsum limits dtt to + or -1000.

Note: In all formulas where it appears, val is the resolved value of the expression to which the function is applied. The val is considered to be counter data if more than one column appears in the expression or if a single-column expression is of counter data type. The val is considered to be gauge data only if the single-column expression is of gauge data type.

TRENDsum By-Variables

A by-variable indicates a time period or a table element such as dsi_key_id. At least one by-variable is required to specify the aggregation level. If both a table element and a time by-variable are given, the time by-variable must be the last one specified.

If a time by-variable is not specified, TRENDsum aggregates the selected data into one row within any element by-variable specified.

Each non-time by-variable must exist as a column in the source data or key table. Each non-time by-variable after the first one must also exist as a column in the destination key table. (The first by-variable is always assigned to dsi_target_name.) If the column name differs between the source and destination tables, use the alias feature. For example: **`by_variable:h_key=frcircuitnumber`**. In this example, the name of the column in the source table is frcircuitnumber and the name of the column in the destination key table is h-key.

Null Columns

You can specify a percentage of source table rows that must be null for TRENDsum to include a null value in the summary row of the output table. See the **-n** command line option. If you omit the **-n** option, the default is 50 (percent).

Rolling Baseline Table

Use command line option **-y** to specify the number of days in the rolling baseline period. (If you specify the **-y** option, the **-e** option is not required.) Once the summarization process has started, you cannot change the name of the source table used for the rolling baseline or the number of days in the baseline period.

Use of the **-y** option places restrictions on the destination table name. This name always begins with SD42 (assuming a 42-day baseline). Normally, the remainder of the name is just the source table name. For example, when the source table name is SDifexceptions the destination table name becomes SD42SDifexceptions.

You may use the **-e** option to append the specified value to the name. For instance, the following option string, **`-y42 -t SDifexceptions -e fore`**, yields SD42SDifexceptionsfore as the destination table name.

Alternatively, you may use the **-a** option to specify a name to use instead of the source table name. Here, the option string, **`-y42 -t SDifexceptions -a mytable`**, yields SD42mytable as the destination table name.

Use of the **-e** option and the **-a** option on the command line at the same time generates the following error message: `Use of baseline name precludes use of destination name`.

A rolling baseline is inherently a daily summary. Therefore, no by-variable greater than **day** is accepted. However, you can specify a lower-level summarization level such as **hour_of_day** to get 24 summary rows of data per day. You can omit the **day** by-variable if you specify the **-y** option. In this case, **day** is automatic.

CAUTION

When using the optional day-of-week or day-of-week-by-hour summarization level, it is strongly recommended that the number of days for the baseline period be evenly divisible by seven; otherwise, some summary rows will summarize a different number of days than others. For example, one summary row might be for seven Mondays while another row might be for six Tuesdays.

Input File Description

The preferred method for providing parameters to TRENDsum is to use an input file. The input file includes statements with keywords that correspond to specific options on the command line. The following keywords introduce these statements:

Keyword	Corresponding Command-Line Option
source_table: *	**-t**
destination_table: *	**-e**
column:	**-c**
by_variable: *	**-b**
baseline days:	**-y**

Keyword	Corresponding Command-Line Option
null percentage:	**-n**
replace table:	**-r** Use of this option is very unusual. If omitted, TRENDsum appends rows to the destination table, which is typically what you want.
first day:	**-w**

* The **by variable**, **source table**, and **destination table** keywords from the 3.5.x and earlier versions are still available. Please note that they may be discontinued in future releases.

Examples

Example1

Execute the following command to invoke TRENDsum to read input from a rate table named **Rnterface_base_** and create a rolling baseline output table named **SD42Rnterface_base_if** based on a 42-day rolling baseline period:

```
$DPIPE_HOME/bin/trend_sum -t Rnterface_base_ -e if -y42
-f $DPIPE_HOME/scripts/if_baseline_hourofday_volume_interface.sum
```

The following file contains the column and by-variable specifications to be included in the **SD42Rnterface_base_if** output table:

$DPIPE_HOME/scripts/if_baseline_hourofday_volume_interface.sum

The contents of this script file are:

```
# trend_sum proc to update hour of day baseline table for
# combined network volume
column:thru_put=(ifinoctets+ifoutoctets):per95,std,wav
```

```
column:volume=(ifinoctets+ifoutoctets):tot,avg
by variable:keyid
by variable:hour_of_day
```

The **SD42Rnterface_base_if** output table contains 24 rows (one row for each hour of the day) for each keyid.

Example 2

Execute the following command to invoke TRENDsum to read a daily summary table named **SDifexceptions** and create a rolling baseline output table named **SDifexceptionsfore** with a 42-day rolling baseline period:

```
$DPIPE_HOME/bin/trend_sum -t SDifexceptions -e fore -y42
-f $DPIPE_HOME/scripts.if_forecast.sum
```

The file **$DPIPE_HOME/scripts.if_forecast.sum** contains the column and by-variable specifications to be included in the **SDifexceptionsfore** output table:

```
# application=interface-reporting;version=3.5;date=1998/06/11
# Copyright © 1998 Desktalk Systems, Inc. All rights reserved.
# trend_sum proc to create days to threshold calculations from
# the daily summary table
column:utilcurrent=P95utilization:per95
column:inutilcurrent=P95waninutil:per95
column:oututilcurrent=P95wanoututil:per95
column:errorcurrent=AVGerror_pct:avg
column:discardcurrent=AVGdiscard_pct:avg
column:utiltrend=P95utilization:dtt[40],f30,f60,f90
column:wanutiltrend=P95wanutil:dtt[40],f30,f60,f90
column:waninutiltrend=P95waninutil:dtt[40],f30,f60,f90
column:wanoututiltrend=P95wanoututil:dtt[40],f30,f60,f90
column:discardtrend=P95discard_pct:dtt[5],f30,f60,f90
column:errorstrend=P95error_pct:dtt[10],f30,f60,f90
by variable:keyid
```

A separate row is created in the **SDifexceptionsfore** output table for each keyid represented in the input table.

A Man Pages

Changes in TRENDsum 3.5.1 from 3.4.x

This section compares 3.4.x and 3.5.1 TRENDsum processing.

1. TRENDsum requires that all (non-time) by variables—except for the first, which becomes dsi_target_name—appear as columns in the destination key table. All components that are used to make up the key are required for possible future TRENDsum operations.
2. The following table points out the TRENDsum formulas in 3.4.x that differ from the formulas used in 3.5.1:

Table A-1: Formulas Used in TRENDsum 3.4.x

Function	**Rate Counter** [1]	**Rate Gauge**	**Summary Item**
tot	same as 3.5.1	$(\Sigma val) / cnt$	same as 3.5.1
avg	$\Sigma val / \Sigma \Delta t$	same as 3.5.1	same as 3.5.1
wav	$\Sigma(val * \Delta t) / cnt$	$\Sigma(val) / cnt$	$\Sigma(val) / cnt$
variance (used in computing std)	$(cnt * (\Sigma(val / \Delta t)^2)$ $-(\Sigma(val / \Delta t))^2)$ $/(cnt * (cnt-1))$	same as 3.5.1	same as 3.5.1
where val is the resolved value of the expression to which the function is applied and cnt is the number of samples being summarized.			

[1] In both TRENDsum 3.4.x and 3.5.1, the val is considered to be counter data if more than one column appears in the expression or if a single-column expression is of counter data type. The val is considered to be gauge data only if the single-column expression is of gauge data type.

3. While the formulas in TRENDsum 3.4.x and 3.5.1 are the same, except for the differences noted in the table above, the results obtained when these formulas are applied to the expressions in a TRENDsum file can be quite different in 3.4.x and 3.5.1. The discrepancy occurs because TRENDsum 3.4.x performs a hidden divide-by-delta_time operation when all functions except tot and cnt

are applied to rate counter expressions. This divide operation is not explicitly specified in the expression, but is performed by the TRENDsum processing code. In 3.5.1, on the other hand, the hidden divide-by-delta_time is not performed.

For example, a 3.4.x expression such as the following:

```
column:inutilization=(((ifinoctets018*8)/
ifspeed013)*100):avg, max
```

needs to be changed as follows in 3.5.1 TRENDsum files to maintain 3.4.x compatibility:

```
column:inutilization=((((ifinoctets018*8)/ifspeed013)/
delta_time004)*100):avg,max
```

You should consider the possibility of obtaining a false max or min if you do not divide a counter expression by delta_time in 3.5.1. For example, assume the poll rate is every 15 minutes and a poll at 1:45 is missed. The following table shows the sample values:

Function	Time	Value of the Counter Expression	Rate-ized Value (value/delta_time) of the Counter Expression	Delta_time
--	1:00	20	0.02222	900
--	1:15	25	0.02777	900
--	1:30	22	0.02444	900
--	2:00	40	0.02222	1800
max		40	0.02777	1800
tot		107	0.09665	4500
avg		26.75	0.02416 (0.09665/4)	1125[1]
wav		29.40	0.02377 (107/4500)	900[2]

[1] Computed as the total delta_time (4500) divided by the number of samples taken (4) = 1125.

[2] Computed as the total delta_time (4500) divided by the number of samples taken plus the sample for the missing period (5) = 900.

With 3.4.x TRENDsum processing, the max function selects the 1:15 sample because of the hidden divide-by-delta_time operation that 3.4.x performs on the counter expression. With 3.5.1, the max function selects the 2:00 sample because the hidden divide operation is not performed. You may consider the 2:00 sample to be a false max in that the sample covers twice as much time as the 1:15 sample.

4. All rate counter expressions (such as utilization) in 3.4.x TRENDsum files need to be adjusted in 3.5.1 because the avg function in 3.5.1 does not divide by delta_time. However, avg calculations are more problematic than min and max calculations due to the time-weighting issue.

   Consider the example in the table above. TRENDsum 3.5.1 calculates the avg function over the 60 minutes as 107/4500 or 0.023777, while TRENDsum 3.4.x calculates the function as 0.09665/4 or 0.02416. The discrepancy results from the weight given the 30-minute sample.

   To remove the discrepancy completely and maintain compatibility in 3.4.x and 3.5.1 avg calculations, you can take one of two actions:

   1. Expressions such as the following in the 3.4.x TRENDsum files:

      ```
      column:inutilization=(((ifinoctets018*8)/
      ifspeed013)*100):avg, max
      ```

      can be changed as follows in 3.5.1 TRENDsum files:

      ```
      column:inutilization=((((ifinoctets018*8)/ifspeed013)/
      delta_time004)*100.0):max
      ```
```
column:inutilization=(((ifinoctets018*8)/ifspeed013)*100.0):avg
```

      Then, you need to adjust the expressions in the .qgr, .qss, and .gos files by multiplying them by (ta_samples/delta_time).

   2. Alternatively, you can replace the avg function in the 3.5.1 TRENDsum files with the wav function, which weights the values based on delta_time in 3.5.1. If you choose this alternative, the TRENDsum files must be changed, the data table columns must be renamed, the col_alias entries

must be updated, and the query files must be updated to reflect the new column names.

In this example, the column avginutilization needs to be changed to wavinutilization and expressions such as the following in 3.4.x TRENDsum files:

```
column:inutilization=(((ifinoctets018*8)/
ifspeed013)*100):avg, max
```

need to be changed as follows in 3.5.1 TRENDsum files:

```
column:inutilization=((((ifinoctets018*8)/ifspeed013)/
delta_time004)*100.0):wav, max
```

If you require the weighted average, alternative a or b produces the desired result. A simpler approach, however, is to use the simple average of value/ delta_time as in 2 above and accept the small discrepancy in the result of calculating the avg function in 3.4.x and 3.5.1.

5. The tot function treats gauge expressions differently in 3.4.x and 3.5.1. In 3.4.x, tot of a gauge averages the values. In 3.5.1, tot totals whatever value it is given regardless of data type. Note, therefore, that the result of tot on a gauge in 3.5.1 is meaningless.

6. In 3.4.x TRENDsum files, you can have avg, max, tot (and perhaps other functions) acting on the same value. If you divide these expressions by delta_time in 3.5.1 TRENDsum files to maintain 3.4.x compatibility, you need to remove tot to a new expression in another line, because you do not want to divide the expression by delta_time when you compute a total. For example, expressions such as the following in 3.4.x TRENDsum files:

```
column:ifinoctets018:avg,max,tot
```

need to be changed as follows in 3.5.1 TRENDsum files:

```
column:ifinoctets=(ifinoctets018/delta_time004):avg,max
column:ifinoctets018:tot
```

trendtimer

Daemon that invokes other TREND processes.

```
trendtimer     [-s schedule_file]
```

Description

trendtimer invokes other TREND applications which need to be run at specific times or regular intervals. It must be run by the user trendadm. trendtimer passes trendadm's environment to each process it starts, including the executable's search path and environment variable values, to insure that the process runs correctly.

trendtimer runs on a 5 minute cycle. Every hour on the hour, and at 5 minute intervals in between, it examines the contents of its schedule file and determines which processes to run at the current time. These processes are run one at a time, in the order in which they appear in the schedule file. Once they are finished, trendtimer waits for the start of the next 5 minute cycle.

trendtimer can be run at any time, but it should be invoked at system startup. The TREND installation adds a command to the system startup file to run /etc/rc.TREND_timer_up, which sets up the TREND environment for trendadm, su's to $DPIPE_HOME/tmp and invokes trendtimer. In the su command in the startup file the entire trendtimer command is quoted to insure that the su command runs the entire trendtimer command line rather than interpreting the trendtimer options as options to su. The simplest way to start trendtimer manually is to become root and execute the command:

```
sh /etc/rc.TREND_timer_up
```

The Schedule File

trendtimer determines what it is going to do by reading a schedule file when it is first invoked. By default, it looks for the file in $DPIPE_HOME/lib/trendtimer.sched. This can be overridden by the -s command line option. The schedule file is a series of statements in this format:

```
<interval> - - <command line>
```

or

```
<interval>+<offset> - - <command line>
```

<interval> is how often the command on this line will be run. It can be either a number of minutes, specified as a simple number MM, or a number of hours and minutes, specified in the format HH:MM. For example, <interval> values of 120 and 2:00 are the same thing. All intervals are counted in <interval> minutes or hours and minutes from unix 0 time, which is midnight, Thursday, January 1, 1970. Intervals chosen should therefore divide evenly into 60 minutes to insure that the process runs at a predictable time. For example, an interval of 5 minutes means the program invoked will run on the hour, and at 5, 10, 15, 20, etc. minutes after every hour because these are 5 minute intervals since midnight, Thursday, January 1, 1970. A program to be run every 2 hours will run at midnight, 2 a.m., 4 a.m., etc.

<offset> tells trendtimer to run the command on this line every <interval> plus <offset> hours and minutes so applications can be run at specific times. It is specified in the format HH:MM where MM must be 00. For example, to run a program every night at 2 a.m., the schedule file entry's time would be 24:00+2:00. Since <interval> is figured from midnight, Thursday, January 1, 1970, 24:00 means every night at midnight. Adding 2 hours to that means every night at 2 a.m. To run exactly at midnight, use 24:00+24:00 (Note that "24:00+0:00" is invalid.). To utilize MM, a time interval of <interval>+<HH:00>+MM is used in the schedule file. The HH value can be a number of hours and the MM value can be a number of minutes.

A special case of <interval> allows a program to be run on a specific day of the week or month. For a specific day of the week:

- For a specific day of the week, use the first two letters of the day of the week (English) on which the command is to be run (SU, MO, TU, WE, TH, FR, SA) instead of the HH:MM <interval> time format. In this format, an <offset> must be used. For example, for a program to run every Saturday night at midnight (i.e., the midnight between Friday and Saturday), the entry is SA+24:00. 24:00 is used because the day of the week abbreviation requires an <offset>, and an offset of 0:00 is invalid. To run every Saturday at 1:00 a.m., the entry is SA+1:00.
- For a specific day of the month, use the letters MONTH*x*, where *x* is the day of the month. An offset is required to specify the hour. For example, MONTH2+7:00 means to run the command every second day of the month at 7:00 a.m. MONTH29+1:00 means to run the command on the 29th day of the month at 1:00 a.m.

By default, trendtimer runs the commands in the schedule file according to local time. You can have it run according to Greenwich Mean Time (GMT) by putting the lowercase letter *g* in front of <interval>+<offset>. This can be useful if you are concerned about daylight savings time.

The hyphens (-) in the syntax are place holders for future arguments. They must be included.

<command line> is the entire command to be run by trendtimer. This is the rest of the line in the file. It does not need to be inside quote marks. Environment variables can be used in the command line by placing them inside “{ }” and omitting the leading $ in the UNIX variable name or leading and trailing % in the Windows variable name.

Comment lines in the file begin with a “#”.

TRENDtimer will only schedule the first 50 entries in the trendtimer.sched file.

A sample trendtimer.sched file is:

```
# trendtimer.sched
5      - - {DPIPE_HOME}/bin/mw_collect -i 5 -n
10     - - {DPIPE_HOME}/bin/mw_collect -i 10 -n
15     - - {DPIPE_HOME}/bin/mw_collect -i 15 -n
```

```
20      - - {DPIPE_HOME}/bin/mw_collect -i 20 -n
60      - - {DPIPE_HOME}/bin/mw_collect -i 60 -n
24:00   - - {DPIPE_HOME}/bin/mw_collect -i 1440 -n
24:00+1:00 - - {DPIPE_HOME}/bin/trendit
24:00+3:00+15 - - {DPIPE_HOME}/bin/keystats
24:00+4:00 - - {DPIPE_HOME}/bin/db_delete_data
24:00+7:00 - - {DPIPE_HOME}/bin/lpr_launch -i 1
MO+7:00 - - {DPIPE_HOME}/bin/lpr_launch -i 7
MONTH2+7:00 - - {DPIPE_HOME}/bin/lrp_launch -i 31
# eof
```

In this example:

- mw_collect is run every 5, 10, 15, 20, and 60 minutes; and every night at midnight for data collection, which is to be run once a day (1440 minutes).
- TRENDit is run every day at 1am.
- keystats is run every day at 15 minutes after 3 a.m.
- db_delete_data is run every day at 4 a.m.
- lpr_launch is run daily at 7 a.m., weekly at 7 a.m. Monday morning, and on the second day of every month at 7 a.m.

Options

-s The path name of the schedule file. The default is to read $DPIPE_HOME/lib/trendtimer.sched.

Examples

To start trendtimer using the schedule file /tmp/mytimer.sched, user trendadm executes the command:

```
trendtimer -s /tmp/mytimer.sched
```

TREND

B Database Administration

The TREND Database and the Sybase SQL Server require special knowledge for proper set up and maintenance. This appendix discusses some of the special issues associated with database administration in TREND.

Estimating the Size of Your TREND Database

While sizing a database is not an exact science, you want to ensure that sufficient space is available to accommodate your raw and aggregated data tables. While 500MB is sufficient for most evaluation purposes, a minimum of 1GB is recommended for data-intensive applications of the TREND software.

To more closely estimate sizing for your TREND database, follow these steps:

1. Estimate the daily raw data table space requirement for each interface collected against:

   (# nodes)(instances per node)(KB per row)(samples per hour)(24 hours)

2. Estimate additional daily table requirements as follows:

   Rate table = Raw table

   Hourly table = Rate table/3

   Daily table = Hourly table/24

   Weekly table = Daily table/7

   Monthly table = Daily table/30

3. Add the daily space requirements for each table to find the total space requirement.

4. Multiply the total space requirement by the retention time specified for each data table in the TREND database (raw, rate, hourly, daily, weekly, and monthly) to find the total space requirement for each data table.

5. Add the total space requirement for all data tables.

6. Multiply the total by 2 to find the recommended TREND database size.

Consider the following collection/aggregation scenario:

Interface Description	**Number of Nodes**	**Instances per Node**	**Size per Row (KB)**	**Samples Collected per Hour**	**Raw Table Size per Hour (MB)**	**Raw Table Size per day (MB)**
PVC miblet	50	5	0.4	12	1.2	29
Cisco interface stats	200	5	0.5	4	2.0	48
etherstats	100	1	0.4	6	0.24	6
TOTAL						83

Furthermore, assume that Raw Tables are to be retained for 2 days, Rate for 7 days, Hourly for 30 days, Daily data for 30 days, and Weekly data for 60 days.

Interface Description	Raw Table Size per day (MB)	Rate Table Size	Hourly Table Size	Daily Table Size	Weekly Table Size	Monthly Table Size	TOTAL
PVC miblet	29	29	10	0.4	0.06	0.01	68.5
Cisco interface stats	48	48	16	0.7	0.1	0.02	112.8
etherstats	6	6	2	0.08	0.01	0.00	14.1
SUBTOTAL	83	83	28	1.18	0.17	0.03	195.4
Retention (days)	2	7	14	60	180	360	
TOTAL	166	581	392	71	31	10.8	1252

The recommended database size in this example is double 1252 megabytes or approximately 2.5 gigabytes.

Enlarging the Database

As your database grows and your sophistication in using it increases, the demands you place on it may require more space. The process of increasing the size of your database involves a series of steps to first allocate the space on your disk, then map that space to the appropriate database.

Note: TREND includes a licensing mechanism to track a purchased TREND database size against the database's daily usage. Initially, you must call DeskTalk Systems Customer Support to obtain a license key for this feature. If daily database usage exceeds the purchased database size, a daily warning message is displayed on the screen and in trend.log. You continue to receive these warning messages until you purchase an expanded TREND database and a new license key is generated.

There are two different databases that may need to be enlarged. One is the TREND database, which Sybase recognizes as “dpipe_db.” You can see how much space is being used by dpipe_db by looking at the Data Manager window.

Note: In general terms, if dpipe_db shows more than 70% of the database as used, it is a good idea to enlarge the database. Other options include deleting unnecessary data or, shortening the length of time you store your data.

The other database that may need more space is the Sybase database tempdb. This is where Sybase creates temporary tables as necessary. If, for example, you run a report that is based upon a database view, Sybase will create a temporary table in tempdb in which to store the data returned by the view.

Sybase installation allocates a small amount of space to tempdb (e.g., 2 MB). In order to use TREND, the size of tempdb must be 30% of the data portion of your dpipe_db database or up to 10% of the largest table in a mature database. See the *TREND Installation Guide* for the procedure to follow to enlarge tempdb.

Once you have installed TREND, you need to enlarge tempdb if you see any of the following messages either in your error log or on your screen:

```
97/02/05 10:18:11.97 server can’t allocate space for object
’syslogs’ in database ’tempdb’ because the ’logsegment’
segment is full. If you ran out of space in syslogs, dump
the transaction log. Otherwise, use ALTER DATABASE or
sp_extendsegment to increase the size of the segment.
```

The example below demonstrates the process of increasing the size of the dpipe_db database by 100%. Because a transaction log is usually 20% the size of its associated database, the example also shows dpipe_db's associated transaction log being increased (by 20 MB).

You can apply the following steps to enlarge any Sybase database. If you are enlarging tempdb, substitute *tempdb* for *dpipe_db*.

1. Log onto Sybase and proceed to the master database:

```
user: su sybase
sybase: isql -Usa -P
1> use master
2> go
```

Note: If you are enlarging tempdb, skip step 2 and step 3 and continue with step 4.

2. Determine the options that are configured on the database:

```
1> sp_helpdb dpipe_db
2> go
```

3. In the output from this command, look for the Name and Status columns. In the Name column, you expect to see the value `dpipe_db`. Assume, for this example, that you see the following configuration options displayed in the Status column:

```
select into/bulkcopy, trunc log on chkpt
```

4. Determine the device numbers that are already being used. Do not use the numbers returned by the following query as values in the vdevno= statements that follow:

```
1> select distinct low/16777216 from sysdevices order by low
2> go
-----------
 0
 1
 2
 11
(4 rows affected)
```

5. Determine the device names that are already being used. Do not use the values returned in the `name` column of the following two commands:

```
1> select name from sysdevices
2> go

name
------------------------------
diskdump
dpipe_disk
dpipe_log
master
tapedump1
tapedump2
(6 rows affected)

1> select name from sysdatabases
2> go

name
------------------------------
dpipe_db
master
model
tempdb
(4 rows affected)
```

6. Next, create space for an extra database device. The variables used in the command are:

name — A name not already used by Sybase. (See the **`select name from sysdevices`** statement in step 5 for the names already in use.)

physname — Pathname of the new database file. The file must not yet exist (i.e., and must not be on an NFS-mounted device.

vdevno — An unused device number. (See the **select distinct low** statement in step 4 for the devices already in use.)

size — <num MB to add>*512

If you are enlarging tempdb, substitute **`tempdb2`** for **`dpipe_db2`** and **`tempdb2.dat`** for **`dpipe_db2.dat`** in the command:

```
1> disk init
2> name=”dpipe_db2”,
3> physname=”/usr300/Dpipe_db/dpipe_db2.dat”,
4> vdevno=6,
5> size=51200
6> go
```

Note: If you are enlarging tempdb, skip step 7.

7. Create the space for the extra log device. The variables are similar to the ones in the previous disk init command.

```
1> disk init
2> name=”dpipe_log2”,
3> physname=”/usr300/Dpipe_tlog/dpipe_tlog2.log”,
4> vdevno=7,
5> size=10240
6> go
```

8. Map the newly created space to the database being enlarged. The variables used in the command are:

database name	Name of the database being enlarged. (Refer to the name in the **select name from sysdatabases** query in step 5.)
on <name from disk init command>	Size of additional file in MB for that name.

If you are enlarging tempdb, change the following command to read simply: **alter database tempdb on tempdb2=xxx**:

```
1> alter database dpipe_db
2> on dpipe_db2=100 log on dpipe_log2=20
3> go

Extending database by 51200 pages on disk dpipe_db2
Extending database by 10240 pages on disk dpipe_log2
```

Note: The following step does not apply for tempdb.

9. Reset the required options on the database. This step is only required if these options were originally set on the database being enlarged (See the output from the **sp_helpdb** command issued in step 2. In this example, dpipe_db and tempdb require that you set the **select into/bulkcopy** option. If you are enlarging tempdb, substitute **tempdb** for **dpipe_db** in the command:

```
1> sp_dboption dpipe_db, “select into/bulkcopy”, true
2> go

Run the CHECKPOINT command in the database that was
changed.
(return status = 0)
```

Note: Skip step 10 and step 11 if you are enlarging tempdb.

10. Reset the trunc. log on chkpt. option for dpipe_db. This step is only required if this option was not originally set on the database being enlarged:

```
1> sp_dboption dpipe_db, “trunc. log on chkpt.”, true
2> go

Run the CHECKPOINT command in the database that was
changed.
(return status = 0)
```

11. Run the checkpoint command for dpipe_db after performing step 9 or step 10:

```
1> use dpipe_db
2> go

1> checkpoint
2> go
```

12. You can see the enlarged database size information from inside Sybase by typing the following commands. (If you enlarged tempdb, substitute **tempdb** for **dpipe_db**.):

    ```
    unix: isql -Udsi_dpipe -Pdsi_dpipe
    1> sp_helpdb dpipe_db
    2> go
    ```

13. Restart TREND:

    ```
    unix: su Password:
    root: sh /etc/rc.TREND_DB_up
    root: sh /etc/rc.TREND_timer_up
    ```

Starting Sybase from the Command Line

To start Sybase from the command line, execute the following commands:

```
unix: su sybase
sybase: cd ~sybase/install
sybase: ./startserver -f ./RUN_<HOSTNAME>_SYBASE > & /dev/null &
```

where <HOSTNAME>_SYBASE is the name of your Sybase SQL Server.

```
or:
```

```
su root
sh /etc/rc.TREND_DB_up
```

For NT, make sure that the service for Sybase is started. (See the *TREND Installation Guide* for details.)

Recovering from Sybase Error 1105

Sybase error 1105 occurs when Sybase is unable to allocate space for a database. The error message returned is:

```
00: 94/07/13 04:07:08.18 server: Error: 1105, Severity: 17,
State: 3

00: 94/07/13 04:07:08.47 server: Can’t allocate space for
object ’syslogs’ in database ’dpipe_db’ because the
’logsegment’ segment is full. If you ran out of space in
Syslogs, dump the transaction log. Otherwise, use ALTER
DATABASE or sp_extendsegment to increase the size of the
segment.
```

This message may be repeated.

If the State value is 1, your database is out of space and needs to be enlarged or cleaned out.

The error message may be accompanied by messages similar to the following:

```
00: 94/07/13 04:07:08.75 server: Automatic checkpointing is
disabled in database ’dpipe_db’ because the log is out of
space. It will continue when the DBO successfully
checkpoints the database. Please free up some space or
extend the database and then run CHECKPOINT.
```

Perform the following steps to free up space in the log.

1. Determine approximately how many pages the transaction log occupies using the commands:

   ```
   1> use dpipe_db
   2> go

   1> select data_pgs (8,doampg)
   2> from sysindexes where id=8
   3> go
   ```

where 8 is the ID for syslogs. The result is the number of data pages (2K pages on most operating systems) that the transaction log occupies. Note that the query results may be inaccurate by as many as 16 pages, but using the query is much faster than counting the rows in the transaction log.

2. Dump the inactive portion of the transaction log using the dump transaction command with truncate_only option. If this command also fails with the 1105 error, run dump transaction with no_log.

3. Repeat step 1. If, however, there are still a large number of pages in the syslogs table, an outstanding transaction is probably preventing the log from being cleared. If this is the case, check syslogshold for old transactions for the database for which the error occurred (in this case it is dpipe_db):

```
1> use master
2> go

1> select * from syslogshold
2> where dbid = db_id(“dpipe_db”)
3> go
```

Determine whether the oldest active transaction can be terminated; it may have been left inactive intentionally. Continue this procedure until there are no other old transactions that can be terminated.

You can clear a long-running transaction in one of two ways:

- Using the Transact-SQL kill command.
- Restarting SQL Server.

If the long-running transaction is due to a runaway query, and the process with the open transaction has been identified, use the kill command to stop the process. This clears the transaction and allows the log to be truncated. If the kill command cannot stop the process, restart SQL Server to resolve the problem.

Restarting SQL Server causes the database to go through normal recovery, so any outstanding transactions are either committed or rolled back.

a. Enter the following commands to restart Sybase SQL Server:

```
unix: su sybase
sybase: cd ~sybase/install
sybase: ./startserver -f ./RUN_<HOSTNAME>_SYBASE > & /dev/null &
```

where <HOSTNAME>_SYBASE is the name of your Sybase SQL Server.

or:

```
su root
sh /etc/rc.TREND_DB_up
```

Note: For NT, make sure that the service for Sybase is started. (See the *TREND Installation Guide* for details.)

Maintaining the Error Log

Sybase writes messages to the error log each time the system is booted and each time a Sybase error is encountered. The name of the error log file on UNIX systems is:

- $SYBASE/install/*hostname*_SYBASE.log for Sybase 11.9.2.

On NT systems, the name of the file is $SYBASE/install/errorlog.

To prevent the file from becoming unnecessarily large, you should periodically delete all messages in the log file preceding the ones written from the beginning of the last boot sequence.

Fixing a Suspect Database

A suspect database is often caused by incorrect ownership of the directory containing the database file and of the database file itself. This can happen if the user installs dpipe_db and/or dpipe_tlog in the DPIPE_HOME directory, or in a directory owned by another application (e.g., CiscoWorks). To fix the problem, change the ownerships of the subdirectory containing the database file and the database file itself to "sybase".

Once this is done, you need to have Sybase reset the database status from suspect back to normal. To do this, follow these steps:

1. Become the sybase user and login to Sybase as the system administrator. In this example there is no password:

   ```
   su sybase
   Password:
   sybase: isql -Usa -P
   ```

2. Allow updates to the system databases.

   ```
   1> sp_configure “allow updates”, 1
   2> go
   ```

   ```
   1> use master
   2> go
   ```

   ```
   1> begin tran
   2> go
   ```

   ```
   1> update sysdatabases
   2> set status = status & ~256
   3> where name = “dpipe_db”
   4> go
   (1 row affected)
   ```

3. If the previous command returns anything other than `1 row affected`, type:

   ```
   1> rollback tran
   2> go
   ```

   and start again from the **`begin tran`** command.

   If the command does return the message `1 row affected,` continue with the following command to commit the transaction:

```
1> commit tran
2> go
```

4. Disallow updates to the system tables:

```
1> sp_configure “allow updates”, 0
2> go
(return status = 0)

1> checkpoint
2> go
```

5. Now, shutdown Sybase.

```
1> shutdown
2> go

Server SHUTDOWN by request. The SQL server is terminating
this process.

CT-LIBRARY error:
ct_results ( ): Network packet layer: internal net library
error: Net-Library operation terminated due to disconnect.

CT-LIBRARY error:
ct_cancel ( ): Network packet layer: internal net library
error: Net-Library operation terminated due to disconnect.
```

6. Now, restart Sybase:

```
unix: su sybase
sybase: cd ~sybase/install
sybase: ./startserver -f ./RUN_<HOSTNAME>_SYBASE > & /dev/null &
```

where <HOSTNAME>_SYBASE is the name of your Sybase SQL Server.

or:

```
su root
sh /etc/rc.TREND_DP_up
```

Note: For NT, make sure that the service for Sybase is started. See the *TREND Installation Guide* for details.

Backup Server

Dumps and loads are performed by the Backup Server program, which runs on the same machine as SQL Server. You can perform backups over the network, using one Backup Server on a remote workstation and another on the local workstation.

Refer to the *SYBASE SQL Server System Administration Guide* for additional information regarding the use of the Backup Server.

On the NT platform, a backup server is automatically created during Sybase installation.

On the UNIX platform, however, you must use an additional utility to install a backup server. The utility depends on the version of Sybase you are running:

- For Sybase 11.9.2, use srvbuild. See “Installing a Backup Server for Sybase 11.9.2 Systems” on page B-15.

Installing a Backup Server for Sybase 11.9.2 Systems

When you install Sybase, a backup server named SYB_BACKUP is created by default. If you want to create a backup server with a different name, use the srvbuild utility on UNIX systems. Follow this procedure:

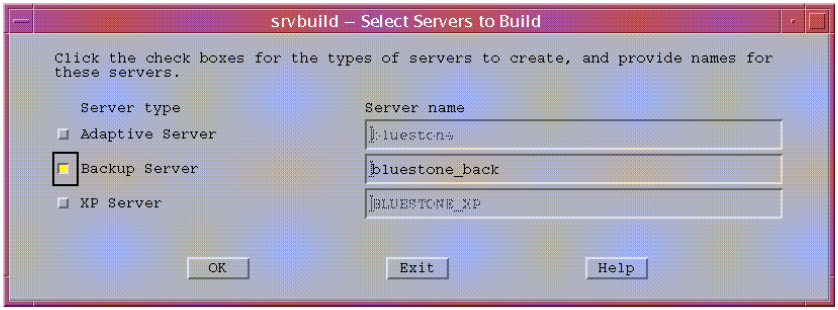
1. Log in as root.
2. Bring up a terminal window.
3. Ensure that the $SYBASE variable points to the directory in which Sybase is installed.
4. Change to the directory in which Sybase is installed:

   ```
   cd $SYBASE/bin
   ```

5. Invoke the srvbuild utility:

   ```
   ./srvbuild
   ```

The Select Servers to Build window appears:

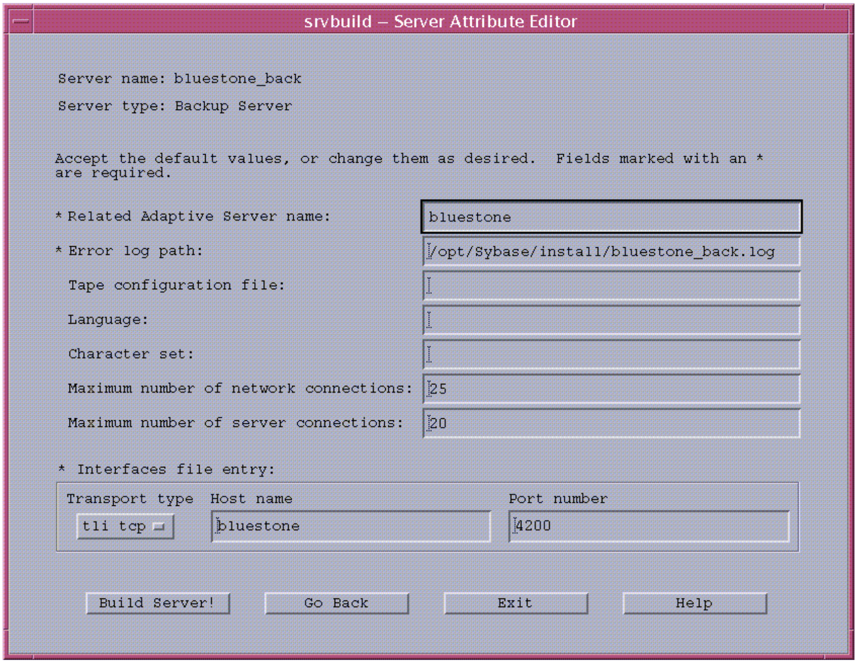


Click the check boxes for the types of servers to create, and provide names for these servers.

Server type	Server name
□ Adaptive Server	bluestone
☑ Backup Server	bluestone_back
□ XP Server	BLUESTONE_XP

6. Select Backup Server as the server type. The default name of the backup server appears in the Server Name list box. We recommend that you accept the default name.

7. Click OK. The Server Attribute Editor window appears:



Server name: bluestone_back

Server type: Backup Server

Accept the default values, or change them as desired. Fields marked with an * are required.

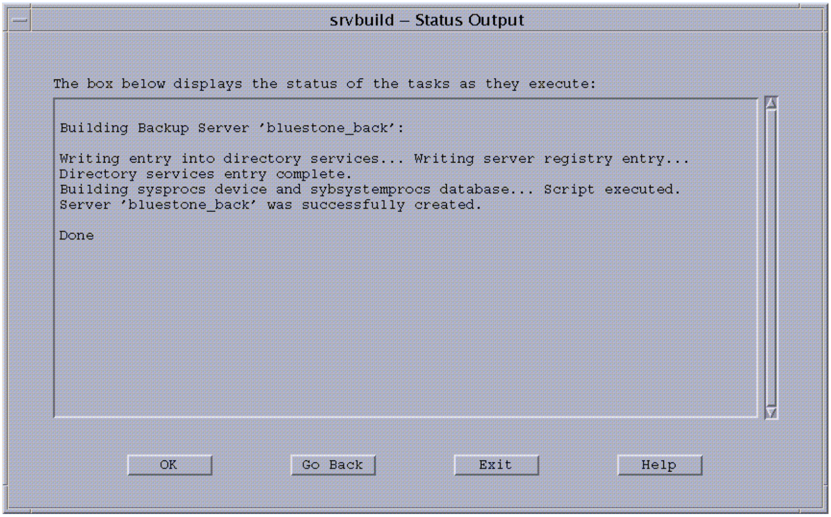
* Related Adaptive Server name:	bluestone
* Error log path:	/opt/Sybase/install/bluestone_back.log
Tape configuration file:	
Language:	
Character set:	
Maximum number of network connections:	25
Maximum number of server connections:	20

* Interfaces file entry:

Transport type	Host name	Port number
tli tcp	bluestone	4200

Default values for the required fields are filled in for you. We recommend that you accept the defaults.

8. Click the Build Server! button. The Status Output window appears. When the build is complete, the word `Done` is displayed in the window.

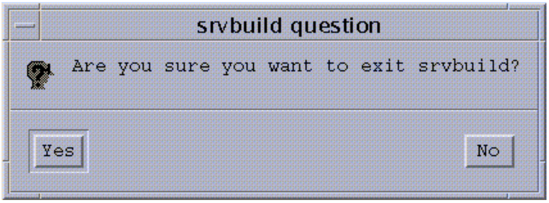


srvbuild – Status Output

The box below displays the status of the tasks as they execute:

```
Building Backup Server 'bluestone_back':

Writing entry into directory services... Writing server registry entry...
Directory services entry complete.
Building sysprocs device and sybsystemprocs database... Script executed.
Server 'bluestone_back' was successfully created.

Done
```

9. Click OK. The srvbuild question window appears:



srvbuild question

Are you sure you want to exit srvbuild?

10. Click `Yes` to close the window and terminate the srvbuild utility.

After you create the new backup server, you need to drop the entry for the old backup server from the sysservers table and add an entry for the new backup server. Follow this procedure:

1. Invoke the isql utility with the following command:

   ```
   isql -Usa -P
   ```

2. Drop the entry for the old backup server (named SYB_BACKUP in this example) by entering the following directive:

   ```
   1> sp_dropserver “SYB_BACKUP”
   2> go
   Server dropped.
   (return status = 0)
   ```

3. Add an entry for the new backup server (bluestone_back in this example) by entering the following directive:

   ```
   1> sp_addserver “SYB_BACKUP”, null, “bluestone_back”
   2> go
   Adding server ‘SYB_BACKUP’, physical name ‘bluestone_back’
   Server added.
   (return status = 0)
   ```

The remainder of the appendix applies to UNIX and NT platforms.

Backing Up the SQL Server Installation

After you have successfully configured a new backup server, use the dump database command to back up your new upgraded SQL Server installation.

To shut down a backup server, specify the backup server name with the Transact-SQL **shutdown** command:

1. Use isql to log into a SQL Server account with System Administrator privileges:

   ```
   isql -Usa -Ppassword -Sserver_name
   ```

2. Enter the following command to shut down the specified backup server:

   ```
   1> shutdown SYB_BACKUP
   2> go
   ```

When you use the shutdown command to stop a backup server, specify the svrname listed in sysservers for Backup Server. SQL Server’s default svrname is SYB_BACKUP.

Creating a Dump Device

The syntax used to create a dump device is:

```
sp_addumpdevice “tape”|”disk”, devicename, “physicalname”, size
```

Examples

1. Create a dump device to a tape:

   For UNIX:

   ```
   isql -Usa -P
   1> sp_addumpdevice “tape”, tapedump, “/dev/rmt/0mn”, 200
   2> go
   ```

   For NT:

   ```
   isql -Usa -P
   1> sp_addumpdevice “tape”, tapedump, “\\.\tape0”, 200
   2> go
   ```

2. Creating a dump device to a disk:

   For UNIX:

   ```
   1> sp_addumpdevice “disk”, diskdump, “/export/home/diskdump.dat”, 2
   2> go
   ```

   For NT:

```
1> sp_addumpdevice “disk”, diskdump, “C:\HOME\diskdump.dat”, 2
2> go
```

In the first example, the tape device must be a non-rewinding device and the size parameter is mandatory. This example specifies a size of 200MB; the actual value should be just larger than your actual database size.

The second example shows a disk dump device. For disk dumps, the size parameter is ignored, but must be included. 2 MB is entered here.

Note: You can use Sybase Central in a Windows environment as an alternative to entering the above commands.

Dumping the Database

The following example illustrates the procedure used to dump a database to a pre-defined dump device.

```
isql -Usa -P
1> dump database dpipe_db to tapedump with init
2> go
```

or:

```
1> dump database dpipe_db to diskdump with init
2> go
```

The init option is recommended, as it overwrites any existing data on the destination dump device.

Example of a Backup Script

The following example shows a sample script with the corresponding procedure for backing up the database. The script contains the commands to perform the backup;

it can be regularly scheduled using a scheduler program such as cron on UNIX or At on NT systems.

First, decide if you want to back up smaller databases such as *master* or *sybsystemprocs* to a filesystem rather than a tape device. Perform either of the following sets of steps (step A or step B) for each database:

A. Use fixed diskfiles defined via sp_addumpdevice:

   1. Add a new **setenv** line for defining the variable name to use an existing Sybase dump device that utilizes a disk file such as:

      ```
      setenv MASTER_DEVICE masterdump
      ```

   2. Change **${TAPE_DEVICE}** on the respective dump command to be the new variable name such as:

      ```
      dump database master to ${MASTER_DEVICE} file=
      "master.dmp" with init
      ```

B. Use diskfiles defined at run-time:

   1. Change **${TAPE_DEVICE}** on the respective dump command to be the full path name such as:

      ```
      dump database master to ’/Full_Pathname/Trend_Backups/
      master.dmp’ with init
      ```

      If you desire to have multiple backup copies available at any instance, you may create these files with a date-designation by using the syntax:

      ```
      dump database master to ’/Full_Pathname/Trend_Backups/
      master_‘date +%y%m%d%H%M‘.dmp’ with init
      ```

   2. Remove the segment starting with **file=** and ending with the double quote (") from the respective dump command.

If you modify the **dump** command for the master database, it is important to keep the **with init** segment.

Second, make sure that the proper access permissions exist on the $SYBASE/install directory for the script to create the $DSQUERY.log.

Third, review the script's log file, `$SYBASE/install/$DSQUERY.log`, on a regular basis to identify any error situations that may arise for a specific database.

Fourth, verify the backup server exists and is running before invoking the script. See "Backing Up the SQL Server Installation" on page B-18.

Note: If all the databases result in a tape volume change, then the system will be waiting.

The script follows and its name is BackupTrendDB.csh.

Note: Remember to replace the parameters identified by the angle brackets (<>) with the requested information. Make sure that the `bin/isql` line has all of its arguments on the same line.

```
#/bin/csh -f
#
# script to backup (i.e. dump) TREND and Sybase databases.
#

setenv DPIPE_HOME <TREND-Directory>
setenv TAPE_DEVICE <Sybase-Name-of-Tape-Device>

# Setup correct Trend environment.
#
source $DPIPE_HOME/lib/Cshrc

# Log data to Sybase log file.
#
echo -n "Starting Sybase/TREND backup on " | tee -a $SYBASE/install/
$DSQUERY.log
date | tee -a $SYBASE/install/$DSQUERY.log

# Start isql session to perform the backups
#
$SYBASE/bin/isql -Usa -P << EOFEOF | tee -a $SYBASE/install/$DSQUERY.log
checkpoint
```

```
go
dump database master to ${TAPE_DEVICE} file="master.dmp" with init
go
dump database sybsystemprocs to ${TAPE_DEVICE} file="sybsystem-
procs.dmp"
go
dump database dpipe_db to ${TAPE_DEVICE} file="db_dpipe.dmp"
go
quit
EOFEOF

echo -n "Completed Sybase/TREND backup on " | tee -a $SYBASE/install/
$DSQUERY.log
date | tee -a $SYBASE/install/$DSQUERY.log
echo " " | tee -a $SYBASE/install/$DSQUERY.log

exit
```

Remember to set the proper execute permissions, such as 755 typically. Place the script into the $DPIPE_HOME/scripts directory. Make sure that the scheduler, such as cron, has the proper environment to know about $DPIPE_HOME and has permission to access this directory.

Loading a Database

The following example illustrates the simplest procedure for restoring a database from a predefined dump device:

```
isql -Usa -P
1> load database dpipe_db from tapedump
2> go
```

where **tapedump** is a tape device, see sp_helpdevices

or:

```
1> load database dpipe_db from diskdump
2> go
```

where **diskdump** is a dump device, see sp_helpdevices, or a file

```
1> online database dpipe_db
2> go
```

Note: Transaction Loading is not discussed here since it is not used.

Installing a TREND Database On a Raw Device

Instructions for installing Sybase on a raw device can be found in the *TREND Installation Guide*. This procedure includes installing the TREND database.

Shutting Down Sybase

To shut down Sybase, enter the following commands:

```
isql -Usa -P
1> shutdown
2> go
```

C Performance Tuning Issues

Note: You can download the scripts mentioned in this appendix from the Desktalk Systems FTP server. Call DeskTalk Systems Customer Support for instructions.

Many factors can positively or negatively affect TREND performance. These factors can be grouped into four functional configuration areas:

- Hardware
- Operating System
- Sybase configuration parameters
- TREND processing options

This document discusses each area. Note, however, that not all identified areas apply to each installation. Factors such as platform or workload may also influence the overall effectiveness of the configuration area being discussed.

Hardware Considerations

TREND is a database application that is designed to run on one to many Sybase databases. The hardware factors that affect overall performance are:

- Processor type and clock speed
- Number of processors
- Amount of RAM
- Disk drive interfaces
- Number of physical disk devices

Processor Type and Clock Speed

Depending on the selected platform, a variety of CPU chips are available for the various system configuration options. For example, a Windows NT system can run on a variety of processors including the following:

- Intel 386, 486, Pentium, Pentium Pro, Pentium II
- AMD K5 and K6
- Cyrix

Note: TREND for Windows NT only runs on Intel-compatible processors, not on Alpha, MIPS, or Power/PC.

Similar considerations apply to Sun Solaris, HP/UX, and IBM AIX platforms.

Factors such as the number of managed elements, collection frequency, and data retention period affect the selection of an appropriate processor for the desired TREND implementation.

In general, on slow processors, the processor is the major performance bottleneck But, as the speed of the processor increases, disk and network I/O becomes the limiting factor for the overall speed of TREND processing.

We recommend that you use the fastest available processor for the selected platform, so that disk and network I/O become the limiting factor.

Number of Processors

Where possible, employ a multiprocessor configuration (two or more CPUs), because TREND and Sybase can be configured to take advantage of multiprocessor environments and run parallel (concurrent) tasks. This decreases the overall run time. In most single system configurations, two or more slow processors will run faster than a single fast processor. For example, a dual Pentium 166 MHz (running Windows NT) provides more throughout than a single Pentium Pro 200MHz or Pentium II 233MHz (with proper Sybase and TREND configuration).

RAM

To get the best performance out of Sybase, you should obtain as much RAM as possible on each processor, regardless of platform. Sybase uses RAM mainly for disk cache and procedure cache buffers. The more available disk cache the less I/O a system does; thus, performance improves.

A guideline for selecting the amount of RAM for a given TREND system follows.

System RAM (per Processor)	**Application**
256 - 512 MB	Small to medium installations (1,000-4,000 interfaces)
512 MB - 1 GB	Medium to large installations (4,000-10,000 interfaces)
1 GB - n GB	Large installations (10,000+ interfaces)

However, this table is just a guideline. You cannot give a system too much RAM. When in doubt, get more RAM than you think you need.

Disk Drive Interfaces

If a system is configured with enough RAM, the next likely bottleneck is disk I/O. Choose an appropriate disk interface (controller), if the platform supplier provides one as an option, to ensure that data can move quickly from the disk to RAM for processing. In general, SCSI interfaces outperform IDE or Enhanced IDE interfaces.

There are at least three flavors of SCSI:

- SCSI-II, 10MB/s burst transfer rate
- Ultra SCSI, 20MB/s burst transfer rate
- Fast Wide Ultra SCSI, 40MB/s burst transfer rate

As with RAM, the faster the interface the faster data can be moved on the I/O bus to RAM for processing. (Do not confuse transfer rate with access time, which deals with disk actuator positioning on the drive itself).

Faster interfaces also provide greater throughput for systems that have multiple I/O devices connected to the interface controller. Some platform vendors have optional (hardware) caching controllers that can provide significant performance improvements over noncaching controllers. For larger systems, you can also use multiple interfaces.

Fast Wide Ultra SCSI is the interface of choice for most medium and large installations.

Number of Physical Disk Devices

If the system needs to update multiple files scattered in different locations on the disk subsystem, a small number of disk drives (for example, one) can reduce system performance considerably. In this case, the disk actuator must perform numerous seeks. A seek operations means a movement of the disk heads, which is the slowest disk operation. Limiting the number and distance of seek operations substantially improves throughput of disk-I/O-bound systems.

In general, the more disk drives the better throughput, assuming that critical files are strategically placed on these drives. For example, if all data that is used for a normally operating system is placed on a single device and the remaining devices are used for backup, no benefit is derived from a multiple disk configuration. When considering the use of multiple disks, make sure the database transaction log is on a separate device from the database itself, if possible. Also, place the operating system's page/swap/virtual memory files on a separate device (or devices).

Some vendors offer disk array products that not only have multiple drives but also provide sophisticated hardware that allows for hardware partitioning of database tables, imbedded cache, disk mirroring, RAID, and other options that improve throughput and data redundancy (high availability). These subsystems are the preferred choice for large installations or for small-to-medium installations where high availability is desired and where hardware budget is not a consideration.

Operating System

Some operating systems have features that need to be enabled to allow Sybase to make the most out of the available operating system resources.

For AIX and HP/UX, asynchronous I/O processing should be enabled to ensure maximum I/O throughput for Sybase. An easy way to tell if asynchronous I/O is enabled for the operating system is to issue an **sp_configure** command from an isql prompt as follows:

```
isql -Usa -P
sp_configure
go
```

The output will have a section that looks something like this:

```
Group: O/S Resources
Parameter Name                Default Memory Used  Config Value  Run Value
------------------------- ----------- -----------  ------------  ----------
max async i/os per engine  2147483647           0    2147483647  2147483647
max async i/os per server  2147483647           0    2147483647  2147483647
o/s asynch i/o enabled              0           0             0           0
```

Notice the line that reads **o/s asynch i/o enabled**. This is a read-only Sybase configuration parameter; you cannot change it. Note, however, that if the run value is zero, asynchronous I/O has not been enabled for the operating system kernel. In this case, ask the appropriate system administrator to enable it. Note that the text may differ slightly depending on the version of Sybase used.

Sybase Configuration Parameters

With Sybase 11 and above, many configuration parameters can be set to improve overall Sybase throughput. The most effective ones are related to memory and cache pools.

When performance tuning any system or application, you change only one parameter at a time. If you change more than one at a time and the net change results in degraded performance, you will not be able to easily identify the change that caused the negative effect.

Here are some suggestions for determining and changing the values of some Sybase configuration parameters:

1. First, obtain a configuration listing from an operational Sybase system by issuing the **sp-configure** command:

   ```
   isql -Usa -P
   sp_configure
   go
   ```

2. Look for the information discussed below in the output.

Total Memory

Find the line for `total memory` in the Cache Manager group in the configuration output listing:

```
Group: Cache Manager
Parameter Name                Default  Memory Used  Config Value  Run Value
------------------------- -----------  -----------  ------------  ---------
memory alignment boundary        2048            0          2048       2048
number of index trips               0            0             0          0
number of oam trips                 0            0             0          0
procedure cache percent            20        32934            20         20
total data cache size               0       126978             0     126978
total memory                     7500       180000         90000      90000
```

By default, Sybase allocates 7,500 pages (15MB) of RAM, which is insufficient for any implementation. In this example, 90,000 2K pages are allocated to Sybase, which amounts to 180MB of RAM allocated to Sybase.

For systems that are dedicated to running TREND, this parameter should be set to 128 MB or at least half of the total RAM in the system. In some cases 60 or 70 percent or more can be allocated to Sybase.

1. To change the total RAM allocated to Sybase:

   ```
   isql -Usa -P
   sp_configure “total memory”,100000
   go
   ```

   Memory is allocated in 2K pages, so the above example results in a 200MB allocation.

2. Restart Sybase to enable the new memory allocation.

CAUTION

Take care when you change this parameter. Some platforms (Sun Solaris, for example) require changes to system memory configuration parameters as well (etc/system).

Furthermore, take care to ensure that client tasks and Sybase do not cause the system to begin paging (thrashing). You can check for this with Performance Monitor under Windows NT or by using sar or vmstat on the various Unix ports.

Procedure Cache Percent

Find the line for `procedure cache percent` in the Cache Manager group in the configuration output listing:

```
Group: Cache Manager
Parameter Name                Default  Memory Used  Config Value  Run Value
---------------------      ----------  -----------  ------------  ---------
memory alignment boundary        2048            0          2048       2048
number of index trips               0            0             0          0
number of oam trips                 0            0             0          0
procedure cache percent            20        32934            20         20
total data cache size               0       126978             0     126978
total memory                     7500       180000         90000      90000
```

The TREND application uses stored procedures to process data. The procedure cache should be at least 20% of the configuration memory. You should use the following formula to calculate the amount of procedure cache required:

1. Multiply the number of user connections (C) by the largest stored procedure size (P), which is typically .3 MB.
2. Multiply that value by an overhead factor of 1.25.
3. Divide that value by the configuration memory (M), which is in 2K pages. (Divide the configuration memory (M) by 500, since there are 2K pages in 1MB, to use the configuration memory value allocated to Sybase in MB.)
4. Multiply by 100 for the percent value.

   Procedure Cache Percent = ((C * P * 1.25) / (M / 500)) * 100

For example, if there are 100 connections, the procedure cache value is 100 * .3 * 1.25 = 37.5 MB. If the configuration memory is 180 MB, which is 90000 / 500, the procedure cache percent is 37.5 / 180 * 100, which is approximately 21%.

If the procedure cache is too small, the stored procedures are fetched from disk rather than read from cache.

1. To change the procedure cache allocation, enter the appropriate value for your installation instead of the value 21 in the following command:

   ```
   isql -Usa -P
   sp_configure “procedure cache percent”,21
   go
   ```

2. Restart Sybase to enable the procedure cache allocation.

Number of User Connections

Find the line for `number of user connections` in the Memory Use group in the configuration output listing:

```
Group: Memory Use
Parameter Name                 Default   Memory Used  Config Value  Run Value
-------------------------- ----------  -----------  ------------  ---------
additional network memory           0            0             0          0
audit queue size                  100           42           100        100
default network packet size       512         #290           512        512
      .
      .
      .
number of user connections         25        10223           128        128
```

1. To change the number of user connections:

   ```
   isql -Usa -P
   sp_configure “number of user connections”,100
   go
   ```

2. Restart Sybase to enable the new user connection allocation.

Choose the number of user connections carefully. Under-allocation results in tasks being unable to connect to Sybase. Too many user connections waste memory that could be allocated to the data cache.

We recommend 100 user connections. This is sufficient in most cases. In cases where RAM is constrained, a more exact calculation may result in additional memory savings. To calculate the number of user connections add up the following:

- Total number of ‘baby’ SNMP collectors (mw_collect -c option). If there is no -c, the default is 5.
- Total number of ‘baby’ RMON collectors (rmon_collect -c option). If there is no -c, the default is 5.
- Total number of concurrent TRENDit processes desired.
- Total number of concurrent TRENDcopy processes desired.
- Total number of concurrent client processes desired.
- Add 5 to cover miscellaneous Sybase processes (for example, bcp, isql).

Cache Pools

Determine your current cache pool configuration by entering the following command:

```
isql -Usa -P
sp_cacheconfig
go
```

The output looks something like this:

```
Cache Name                   Status    Type      Config Value     Run Value
-------------------------- --------- --------  ------------  -------------
default data cache           Active    Default        0.00 Mb      112.11 Mb
                                                 ------------  -------------
                                         Total        0.00 Mb      112.11 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
      Config Size: 0.00 Mb,   Run Size: 112.11 Mb

 IO Size Wash Size   Config Size      Run Size
-------- ---------  ------------  ------------
    2 Kb    512 Kb       0.00 Mb     112.11 Mb
```

By default, Sybase ships with only one data cache called *default data cache*. This cache has only 2K buffers in it.

You should add a 16K buffer pool to the default data cache. The size varies depending on the available memory. For constrained environments, start by allocating about 10 percent of the total available (see Run Size value in the above example). Start increasing the value until a noticeable performance degradation is evident and then back off some. Nonconstrained environments typically can allocate 20-40 percent of the available cache to the 16K buffer pool.

To change (add) a 16K buffer pool of 12MB:

```
isql -Usa -P
sp_poolconfig “default data cache”,”12M”,”16K”
go
```

This change is dynamic; a Sybase restart is not required.

Online Engines

Find the lines for `maximum` and `minimum online engines` in the Processors group in the configuration output listing:

```
Group: Processors
Parameter Name                 Default  Memory Used  Config Value  Run Value
-------------------------  -----------  -----------  ------------  ---------
max online engines                   1          132             1          1
min online engines                   1            0             1          1
```

In multiprocessor environments, you need to configure Sybase to use the additional CPUs. For systems with less than four processors set the max online engines to the total number of processors. In systems with four or more processors, you can gain additional throughput by allowing one or more processors to be available for client processes (for example, TRENDit, mw_collect, rmon_collect, TRENDcopy). For example, on an eight-processor system, allocate six processors to Sybase.

1. To change the available processors:

   ```
   isql -Usa -P
   sp_configure “max online engines”,6
   go
   ```

2. Restart Sybase to enable the new processor allocation.

Transaction Log Cache

In most cases, you should create a separate named cache for the transaction log, because the transaction log is primarily write-intensive. Excessive writing to a shared cache (such as the default data cache) causes unnecessary flushing of pages (to disk) and thrashing or rereading of pages, thus decreasing overall system performance.

To set up a named-cache for the transaction log:

```
isql -Usa -P
sp_cacheconfig “logcache”,”5M”,logonly
go
```

This example creates a 5MB named cache for log pages only (logonly option). Once the cache has been created, you need to bind the transaction log to it.

Note: The database must be set to single-user mode before you can bind the transaction log to the log cache.

To bind the transaction log to the newly created log cache:

1. Place the database into single-user mode as follows (be sure there are no active TREND transactions before proceeding):

```
isql -Usa -P
sp_dboption dpipe_db,”single user”,true
go
```

The output looks something like this:

```
Database option ‘single user’ turned ON for database ‘dpipe_db’.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
```

2. Switch to the dpipe_db database (use dpipe_db) and issue a checkpoint command followed by the sp_bindcache command as follows:

```
use dpipe_db
go
checkpoint
go
sp_bindcache logcache, dpipe_db, syslogs
go
```

3. Switch back to the master database to return the dpipe_db database to its normal (multi-user) state:

   ```
   use master
   go
   sp_dboption dpipe_db,”single user”,false
   go
   ```

   The output looks something like:

   ```
   Database option ‘single user’ turned OFF for database ‘dpipe_db’.
   Run the CHECKPOINT command in the database that was changed.
   (return status = 0)
   ```

4. Checkpoint the dpipe_db database:

   ```
   use dpipe_db
   go
   checkpoint
   go
   ```

5. Allocate a buffer pool to the newly created cache (logcache):

   ```
   isql -Usa -P
   sp_poolconfig “logcache”,”4M”,”8K”
   go
   ```

6. Restart Sybase to enable the new log cache.

Transaction Log Size

Determine the TREND transaction log I/O size by entering the following commands:

```
isql -Usa -P
use dpipe_db
go
sp_logiosize
go
```

The output of this command looks like this:

```
The transaction log for database ‘dpipe_db’ will use I/O size of 2 Kbytes.
(return status = 0)
```

By default, Sybase is configured to use a 4K transaction log. But, Sybase does not allocate any 4K buffers in the default cache configuration for the transaction log to use. Thus, a 2K transaction log is used for systems that have not changed the cache configuration. Sybase recommends a 4K logiosize for user databases (particularly OLTP activity). Some TREND installations fare better with an 8K log I/O size.

To change the log I/O size, allocate a cache pool for only the transaction log (8K for example) by entering the following command:

```
isql -Usa -P
use dpipe_db
go
sp_logiosize “8”
go
```

The output of this command looks like this:

```
The transaction log for database ‘dpipe_db’ will use I/O size of 8 Kbytes.
(return status = 0)
```

Number of Locks

The number of locks has little to do with performance. But, if the total number of locks specified is insufficient, application problems such as the following can ensue:

- Cannot obtain lock
- Failure to connect

Be sure the number of locks is adequate. Over-allocating locks is rarely detrimental. By default, Sybase defines 5,000 locks. Desktalk Systems recommends a minimum of 20,000 locks. Implementations that want to run a large number of baby collectors or parallel instances of TRENDcopy and TRENDit should use 40,000 to 50,000 locks.

1. To change the number of locks:

   ```
   isql -Usa -P
   sp_configure “number of locks”,40000
   go
   ```

   This example sets the number of locks to 40,000.

2. This change is static. Restart Sybase to implement the change.

Binding the Key Tables

Placing all key tables into their own cache can help performance by requiring no page I/O when the key tables are referenced, thereby reducing disk access. There are two necessary phases to implement this functionality; they are creating the named cache and the binding of individual key tables to the named cache.

Creating the Named Cache

The first step is to create a named cache to bind the key tables into such as keycache. It is advisable to implement this functionality after the TREND database has reached its normal size to avoid under- or over-allocating the amount of cache necessary. After you create the cache named keycache, you must shut the Sybase SQL Server down and restart it for the named cache to be visible.

For example, to add the named cache with the name of the cache equal to keycache and the size of cache equal to 1 MB, enter the following commands:

```
isql -Usa -P
1>sp_cacheconfig keycache, “1M”
2>go
```

Remember to shut the Sybase SQL Server down and restart it. See “Cache Pools” on page C-10 on how to configure the cache you just created.

Binding the Individual Key Tables to the Named Cache

The next step is to bind the key tables to the named cache. You can select the key tables of your most frequently used data tables, key tables of your largest data tables, or any key table that would help the performance by reducing disk access. Enter the following commands:

```
isql -Usa -P
1>use dpipe_db
2>go
```

Use the following command to return the names of all data tables with their corresponding key table names.

```
1>select dsi_data_table, dsi_key_table from dsi_key_tables
2>go
```

Once you have identified the names of all the key tables that you wish to cache, repeat the following command for each key table name:

```
1>sp_bindcache  keycache, dpipe_db, "dsi_dpipe.key_tabl_nam"
2>go
```

where *key_tabl_nam* is the name of the key table to cache. Note that you must enter the quotes.

For example, to bind the key tables Kib_ii_ifentry_, Ky_interface_perf_, and K5_frcircuitentry_ to the cache named keycache, enter the following commands:

```
1>sp_bindcache  keycache, dpipe_db, "dsi_dpipe.Kib_ii_ifentry_"
2>go
1>sp_bindcache  keycache, dpipe_db, "dsi_dpipe.Ky_interface_perf_"
2>go
1>sp_bindcache  keycache, dpipe_db, "dsi_dpipe.K5_frcircuitentry_"
2>go
```

Verify the Bindings Were Successful

Issue the following command to verify that the binding of the key tables is successful:

```
isql -Usa -P
1>sp_helpcache
2>go
```

Status should be V (for valid) on each binding. The output will look something like this:

```
Cache Name               Config Size   Run Size   Overhead
-----------------------  -----------   --------   --------
default data cache           0.00 Mb   35.87 Mb    1.93 Mb
keycache                     1.95 Mb    1.95 Mb    0.15 Mb

Memory Available For         Memory Configured
Named Caches                 To Named Caches
--------------------         -----------------
            37.84 Mb                   1.95 Mb

There is 35.89 Mb of memory left over that will be allocated to
the default cache

------------------ Cache Binding Information: -------------------
Cache Name  Entity                                 Type    Status
----------  -----------------------------------    ------  ------
keycache    dpipe_db.dsi_dpipe.Kmon_ethstats_      table   V
keycache    dpipe_db.dsi_dpipe.KDrouter_traffic    table   V
keycache    dpipe_db.dsi_dpipe.Kmon2_alhost_       table   V
keycache    dpipe_db.dsi_dpipe.KDhp_filesystem     table   V
keycache    dpipe_db.dsi_dpipe.Kmon2_almatrix_     table   V
keycache    dpipe_db.dsi_dpipe.Kdbaselineday       table   V
keycache    dpipe_db.dsi_dpipe.KHhp_compsystem     table   V
     .
     .
     .
```

You can perform these functions as necessary:

- Add or modify the named cache, keycache, when you suspect that the cache allocation is out of date (i.e., needs to be smaller or larger).
- Bind new key tables to the named cache.

TREND Processing Options

You can improve throughput on machines with sufficient CPU and RAM resources by running processes in parallel. The notion of parallel processing applies to collection, copying of TREND tables, TRENDit processing, TRENDsum processing, and db_delete_data processing.

In CPU-constrained environments, running multiple processes concurrently may result in an increase in run time and a decrease in throughput because additional processor overhead is associated with the operating system's task management.

DeskTalk systems has created scripts to implement parallel processing functionality (for example, scripts that invoke concurrent instances of a program, each with specific arguments).

Contact DeskTalk Systems, Inc. via e-mail at the following address for information on the availability of scripts for your platform and for download instructions:

support@desktalk.com

Data Collection

The mw_collect and rmon_collect programs have a command line switch (-c) that allows you to control the number of concurrent collection processes. By default, this number is five. You can reduce SNMP and RMON collection cycles by increasing this number, for example, to 25. If you change this number, make sure enough user connections are configured for Sybase as discussed in "Sybase Configuration Parameters" on page C-6.

TRENDcopy

When you use TRENDcopy to copy database tables, there are a variety of processing options. For example, you can copy several tables at a time (thus enabling processor overlap). If a specific table is not specified (with the -t) option, TRENDcopy copies tables of the appropriate processing option serially. This may result in excessive run times. By employing the -t option, you can run multiple TRENDcopy instances simultaneously, thus improving throughput and reducing run times.

TRENDit/TRENDsum

When you use TRENDit or TRENDsum, you can implement a number of processing options with command line switches. In environments that are not resource-constrained, you can reduce elapsed processing time by running multiple instances of TRENDit and TRENDsum in parallel.

For TRENDit, use the -t command line option. You must know and identify all tables that require TRENDit processing.

Since you invoke TRENDsum on a per-table basis, you will know the identity of the tables being processed.

In a typical environment, TRENDit is launched from the trendtimer.sched file located in the $DPIPE_HOME/lib directory. The file looks something like this:

```
# trendtimer.sched
5  - - {DPIPE_HOME}/bin/mw_collect -n -i 5 -c 25 -n
10 - - {DPIPE_HOME}/bin/mw_collect -n -i 10 -c 25 -n
15 - - {DPIPE_HOME}/bin/mw_collect -n -i 15 -c 25 -n
20 - - {DPIPE_HOME}/bin/mw_collect -n -i 20 -c 25 -n
60 - - {DPIPE_HOME}/bin/mw_collect -n -i 60 -c 25 -n
5  - - {DPIPE_HOME}/bin/rmon_collect -n -i 5 -c 25 -n
10 - - {DPIPE_HOME}/bin/rmon_collect -n -i 10 -c 25 -n
15 - - {DPIPE_HOME}/bin/rmon_collect -n -i 15 -c 25 -n
20 - - {DPIPE_HOME}/bin/rmon_collect -n -i 20 -c 25 -n
60 - - {DPIPE_HOME}/bin/rmon_collect -n -i 60 -c 25 -n
24:00+1:00 - - {DPIPE_HOME}/bin/logbackup
24:00+2:00 - - {DPIPE_HOME}/bin/trendit -s -z 30 -f 60
    .
    .
    .
```

You can run parallel instances of TRENDit in a number of ways depending on your requirements. For example, you can add multiple TRENDit lines to the trendtimer.sched file using the -t option:

```
# trendtimer.sched
5  - - {DPIPE_HOME}/bin/mw_collect -n -i 5 -c 25 -n
10 - - {DPIPE_HOME}/bin/mw_collect -n -i 10 -c 25 -n
15 - - {DPIPE_HOME}/bin/mw_collect -n -i 15 -c 25 -n
20 - - {DPIPE_HOME}/bin/mw_collect -n -i 20 -c 25 -n
60 - - {DPIPE_HOME}/bin/mw_collect -n -i 60 -c 25 -n
5  - - {DPIPE_HOME}/bin/rmon_collect -n -i 5 -c 25 -n
10 - - {DPIPE_HOME}/bin/rmon_collect -n -i 10 -c 25 -n
15 - - {DPIPE_HOME}/bin/rmon_collect -n -i 15 -c 25 -n
20 - - {DPIPE_HOME}/bin/rmon_collect -n -i 20 -c 25 -n
60 - - {DPIPE_HOME}/bin/rmon_collect -n -i 60 -c 25 -n
24:00+1:00 - - {DPIPE_HOME}/bin/logbackup
24:00+2:00 - - {DPIPE_HOME}/bin/trendit -t mib_ii_ifentry_ -s -z 30 -f 60
24:00+2:00 - - {DPIPE_HOME}/bin/trendit -t mib_ii_system_ -s -z 30 -f 60
24:00+2:00 - - {DPIPE_HOME}/bin/trendit -t cisco100_lsystem_perf_ -s -z 30 -f 60
    .
    .
    .
```

In considering a parallel implementation, you need to assess the machine's CPU and RAM to ensure that running parallel processes do not have a detrimental performance impact.

db_delete_data

You may want to run several concurrent instances of the db_delete_data program to age records from multiple database tables (thus enabling processor overlap). If you do not specify a particular table (with the -t option), db_delete_data processes tables having the appropriate aging option serially.

This may result in excessive run times. By employing the -t option, which will only perform update statistics and record aging on the specific table, you can run multiple instances of db_delete_data, and thus improve throughput and reduce run times.

Intermediate/Reporting Tables

When CPU and memory constraints are not at issue, the biggest factor that can negatively impact performance is table size. The larger the table the longer it takes to process on any machine with any configuration. Take care to retain only the data in the tables that is required for nightly TRENDit/TRENDsum processing. To this end, an optimal configuration retains data as follows:

Type of Data	**Retention Period**
Raw	1 day
Rate	1 day
Hourly	7 days
Daily	31 days
Weekly	31 days
Monthly	6 months

If you need to retain data longer, you should use TRENDcopy to create an intermediate archive or reporting table.

For example, assume you want to keep hourly mib_ii_ifentry (Hib_ii_ifentry_) data for one month. Use the following TRENDcopy command to create and maintain an intermediate reporting table:

```
trendcopy -t hour_mib-II_ifEntry::hour_mib-II_ifEntry_rpt
-s source_server -S target_server
```

The first invocation of TRENDcopy creates all the necessary data dictionary and table entries and copies the entire source table (in this case, hour_mib-II_ifEntry) to the target table (hour_mib-II_ifEntry_rpt). Subsequent TRENDcopy invocations copy only data from the source table that is newer than the data in the target table.

After you create the reporting table, use the TREND Data Manager to change the retention period of the source table to 7 days and of the target (reporting) table to 31 days. This ensures that TRENDit processes only the minimum amount of data on a nightly basis.

Note that the Raw and Rate tables typically have the most data; then the Hourly tables. The number of rows in the Daily, Weekly, and Monthly tables is comparatively insignificant.

TREND

D Setting Up a Satellite Server Environment

Satellite servers and central servers are installed in the same manner. However, there are numerous ways that they can be configured. The most common, and straightforward, way is to:

1. Set up polling to collect raw data at the satellite servers.
2. Run TRENDit to convert the raw data to delta data.
3. Run TRENDcopy to send the delta data to the central server. A Sybase interfaces file defining the central server must exist on the satellite server (the server that issues the TRENDcopy command). See “Creating the Sybase Interfaces File on UNIX Systems” on page D-8 or “Creating the Sybase Interfaces File on Windows Systems” on page D-15 for details.
4. Run TRENDstep, TRENDsum, and TRENDrank on the central server to create the base and summarization tables for reporting.

Reporting is done at the central server.

An Elan license manager should be installed at each server. When you call DeskTalk Systems Customer Support to establish the keys for each license manager, the support person will give you a license key based on your configuration.

The following sections point out considerations for implementing the server environment described above. The discussion assumes a basic knowledge of TRENDit, TRENDcopy, and TREND architecture.

Data Management on the Satellite Server

If you do not need to generate reports on each satellite server, you do not have to create hourly, daily, weekly, and monthly tables on the satellite server. You only need to acquire delta data from the satellite server. The hourly, daily, weekly, and monthly tables can then be created at the central server.

Use the **-r** option in the **trendit** command that you issue from the trendtimer.sched file to specify that only delta data be calculated at the satellite server. For example:

trendit -s -r -z *error_%*

However, this command calculates raw tables into delta tables serially. If your environment is not resource-constrained, you can save elapsed processing time by running multiple instances of TRENDit concurrently using the -t option. (See "Performance Tuning Issues" on page C-1 for a complete discussion of the performance implications.) For example:

trendit -s -r -z *error_%* **-t** *table_name*

This command calculates delta data for only the table specified by *table_name*. Include a trendit command line (with the -t option) in trendtimer.sched for each raw table from which you want to calculate delta data.

If your servers are set up in a UNIX environment, you can download the following scripts from the DeskTalk Systems FTP site and use them as an alternative to the trendit command:

- TrendIt.sh
- TrendItLoop.sh

To use these scripts:

1. Place the downloaded scripts in the $DPIPE_HOME/bin directory on the satellite server where TRENDit is to execute:
2. Replace the TRENDit commands in the appropriate trendtimer.sched file with one of the following commands:
   - **TrendIt.sh -s -r -z** *error_%* **-t** *table_name* ...

     This command invokes TRENDit to calculate delta data from the raw table specified by *`table_name`* and generates some logging information. Include a TrendIt.sh command line for each raw *table_name* from which you want to calculate delta data.
   - **TrendItLoop.sh -s -r -z** *error_%* ...

     This one command invokes concurrent instances of TrendIt.sh to calculate delta data from all raw tables.

Note: These scripts are available for UNIX environments only. For NT environments, you must use a form of the TRENDit command described above.

Using TRENDcopy

You can use TRENDcopy in one of the following ways:

- To push the data from the satellite server onto the central server. That is, run TRENDcopy on the satellite server according to the scheduling information given in the trendtimer.sched file residing on the satellite server.
- To pull the data from the satellite server onto the central server. That is, run TRENDcopy on the central server according to the scheduling information given in the trendtimer.sched file residing on the central server.

Either method is effective. To implement one of these methods, place the following command in the appropriate trendtimer.sched file:

trendcopy -s *source_server* **-S** *target_server* **-r -R**

This command copies raw and delta tables serially, which may result in excessive run times. (See “Performance Tuning Issues” on page C-1.) To improve throughput and reduce run times, you can execute multiple TRENDcopy processes (with the **-t** option) concurrently. For example:

trendcopy -s *source_server* **-S** *target_server* **-t** *table_name*

This command copies only the table specified by *table_name* to the central server. Include a trendcopy command line (with the -t option) in trendtimer.sched for each delta table you want to copy.

If your servers are set up in a UNIX environment, you can download the following scripts from the DeskTalk Systems FTP site and use them as an alternative to the trendcopy command:

- TrendCopy.sh
- TrendCopyLoop.sh

To use these scripts:

1. Place the downloaded scripts in the $DPIPE_HOME/bin directory on the server where you want to execute TRENDcopy.
2. Replace the trendcopy commands in the appropriate trendtimer.sched file with one of the following commands:
   - **TrendCopy.sh -s** *source_server* **-S** *target_server* **-t** *table_name*

     This command copies only the delta table specified by *table_name* from the source server to the target server and generates some logging information. Include a TrendCopy.sh command line for each *table_name* you want to copy.
   - **TrendCopyLoop.sh -s** *source_server* **-S** *target_server*

     This command invokes concurrent instances of TrendCopy.sh to copy all delta tables from the source server to the target server.

     These scripts also generate some logging information.

Note: The scripts are available for UNIX environments only. For NT environments, you must use a form of the trendcopy command described above.

Timing TRENDit and TRENDcopy Operations

Consider the following items when you schedule TRENDit and TRENDcopy operations:

1. First, to generate delta data for a given day, TRENDit requires at least two different samples whose timestamp is after 1:00 a.m. the next day. Therefore, if the satellite server is getting polled data every 15 minutes, you should schedule TRENDit to run after polling completes, say for 1:45 a.m. (Polling operations do not always take the same amount of time, so be sure to be sure to figure in a

buffer that gives polling time to complete before you schedule TRENDit to run.)

2. Second, each satellite server should perform its delta data copy to the central server only after TRENDit has created the delta tables. Therefore, you must schedule TRENDcopy to execute after local TRENDit processing is complete. (Since the time needed for local TRENDit processing can vary, be sure to figure in a buffer that allows TRENDit processing to complete before you schedule the TRENDcopy start time.)
3. Finally, schedule TRENDstep, TRENDsum, and TRENDrank to run on the central server only after the central server has received all delta data from the satellite servers (that is, after remote TRENDit and TRENDcopy processes are complete).

When TRENDcopy is Delayed

Only one server invokes TRENDcopy, and that server alone is responsible for completing TRENDcopy execution. The server uses the Sybase interfaces file for communication information regarding the other server. Create the Sybase interfaces file on the server where TRENDcopy is executed. See “Creating the Sybase Interfaces File on UNIX Systems” on page D-8 or “Creating the Sybase Interfaces File on Windows Systems” on page D-15, as appropriate, for instructions on creating this file.

If a problem on the satellite server makes delta data unavailable for copying, there are no automated retries. In this case, the copy process is attempted on the next day at the scheduled time, or you can invoke it manually.

If no delta data is copied to the central server, reports are still generated (or attempted), depending on the rollup tables that are used in the reports.   For example, if no data is available for hourly- or daily-based reports, weekly reports containing only the rollup data that was previously collected are generated. However, all data is time-stamped, so the delay in getting data to the central server does not affect its usage when it finally gets there.

Node Lists and Polling Policy

In general, the central server does not need to know about nodes, node types, polling policies, etc., because it only aggregates and reports data that is maintained by unique property (key) table records.

Thus, the recommended approach is for the satellite servers to maintain their separate node lists and polling policies for those node lists, to perform separate current node type discovery, and to perform specific IP-address-range discovery for new nodes (where range on one satellite does not overlap with the range on another satellite).

Then, the TRENDit and TRENDcopy processes, as detailed above, can be configured on each satellite sever to place delta data onto the central server.

If you want to deviate from this approach, discuss your plan with your Desktalk System Engineer or Customer Support person.

Firewall Considerations

Several TREND functions use port numbers for client/server communication. Therefore, consider the following:

1. If you are using a license manager at each server, you do not have to be concerned with the satellite servers' and pollers' requesting tokens through the firewall. However, if you are using a central license manager, which is inside a firewall, you need to know the following information:

   The license manager uses UDP on port 5841 for requests from the remote poller/satellite; the licensing response to this UDP packet from the remote poller/satellite uses a different destination port when it responds to the source port's request. Therefore, your firewall rules have to allow UDP port 5841 from the remote satellite/poller and, depending on your firewall software, allow UDP ports from the license grantor (your central TREND server) back to the satellite/poller.

2. The Sybase interface uses TCP/IP and is configurable. If the database is on UNIX, the port number is defined in the interfaces file ($SYBASE/interfaces). The default port number is 2052.

   If the database is on NT, you can check the port number via the Dsedit utility; the default is 5000.

3. The TRENDweb product line uses TCP/IP. Use the trendweb.cfg file to configure TRENDweb 1.1. Use the RunServer script file to configure TRENDweb 2.0. Use the Configure script to configure TRENDweb 3.0 and above.

4. The TREND collection processes (mw_collect and rmon_collect) use port 161 by default. You can modify the default on a per-collection-process-invocation basis by using the **-P** option (uppercase) on the invoking command.

Creating the Sybase Interfaces File on UNIX Systems

You use different utilities to create the Sybase interfaces file for Sybase 11.0.x., 11.5, and 11.5.1 on UNIX systems as described below. Create the file on the server that runs the TRENDcopy command.

For Sybase 11.0.x

Use the sybinit utility to install the Sybase interfaces file for Sybase 11.0.x on UNIX systems. Perform these steps:

1. As user sybase, bring up a terminal window.
2. Bring up the sybinit utility by typing the following command:

```
$SYBASE/install/sybinit
```

3. Select Edit/View Interfaces File from the main sybinit menu.
4. Select Add a New Entry from the INTERFACES FILE TOP SCREEN.
5. Enter 1 for Server Name, press Return, and then type in the name of the secondary Sybase server (not to be confused with the server's host name) after the prompt. Press Ctrl+a to accept your entry.
6. Select Add a New Listener Service from the SERVER INTERFACES FILE ENTRY SCREEN.
7. Enter 1 for Hostname/Address from EDIT TCP SERVICE, press Return, and then type the secondary server's host name after the prompt.
8. Enter 2 for Port from EDIT TCP SERVICE, press Return, and then type the port number that is configured for Sybase connections on the secondary server after the prompt.
9. Press Ctrl+a to accept your entry.
10. Enter y to the prompt:

    ```
    Is this information correct?
    ```

    and press Return.
11. Press Ctrl+a to accept your entry.
12. Enter y to the prompt:

    ```
    Write the changes to the interfaces file now?
    ```

    and press Return.
13. Press Ctrl+x to exit the current screen.
14. Press Ctrl+x again to exit the sybinit utility.

For Sybase 11.5

Use the sybsetup utility to install the Sybase interfaces file for Sybase 11.5 on UNIX systems. Perform these steps:

1. Bring up a terminal window.
2. Bring up the sybsetup utility by typing the following command:

   ```
   $SYBASE/bin/sybsetup
   ```

3. Select Specify Directory Services Information from the Configure Sybase Servers menu.
4. Highlight Sybase Interfaces File on the Select a Directory Service dialog, verify the location of the interfaces file, and press OK.
5. Highlight the Sybase Server from the list of Available Servers and select Add New Server Entry on the Directory Service Session dialog.
6. On the Server Entry Editor dialog, type in the name of the secondary Sybase server (not to be confused with the server's host name) and select Add New Network Transport.

   This brings up a secondary dialog entitled Network Transport Editor.

7. Select tli tcp as the Transport Type, enter the secondary server's host name, enter the port number that is configured for Sybase connections on the secondary server, and press OK.

   This returns you to the Server Entry Editor dialog.

8. Press OK. This returns you to the Directory Service Session dialog.
9. Press Close Session. This returns you to the Select a Directory Service dialog.
10. Press Exit.
11. Press Yes when prompted about exiting dsedit. This returns you to the sybsetup dialog.
12. Press Exit (the icon that has an arrow through the door).

You can validate the result by viewing the $SYBASE/interfaces file.

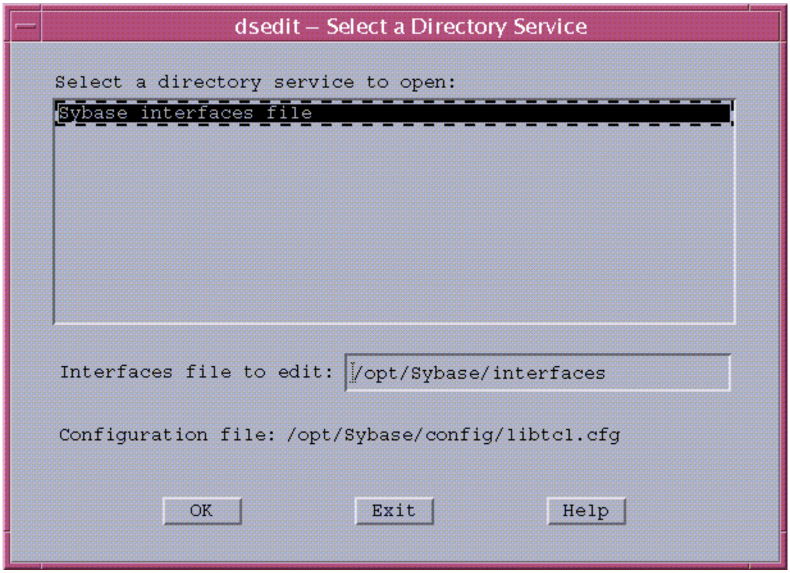
For Sybase 11.5.1

Use the dsedit utility to install the Sybase interfaces file for Sybase 11.5.1 on UNIX systems. Perform these steps:

1. Bring up a terminal window.
2. Bring up the dsedit utility by typing the following command:

   $SYBASE/bin/dsedit

   The Select a Directory Service dialog appears:



dsedit – Select a Directory Service

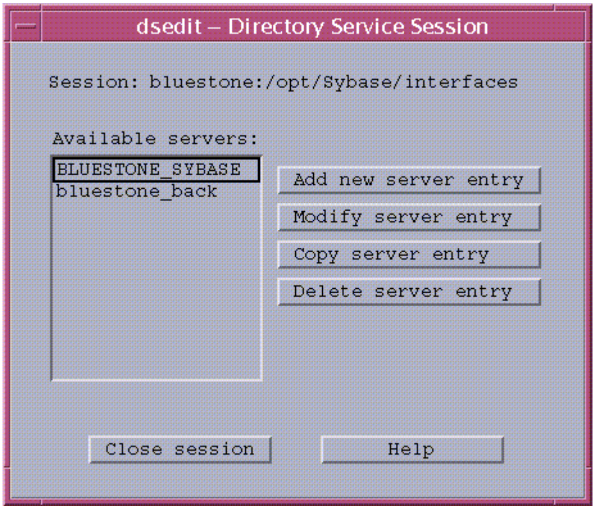
Select a directory service to open:

Sybase interfaces file

Interfaces file to edit: /opt/Sybase/interfaces

Configuration file: /opt/Sybase/config/libtcl.cfg

3. Highlight the Sybase Interfaces File on the Select a Directory Service dialog, verify the location of the interfaces file, and click OK. The Directory Service Session dialog appears:



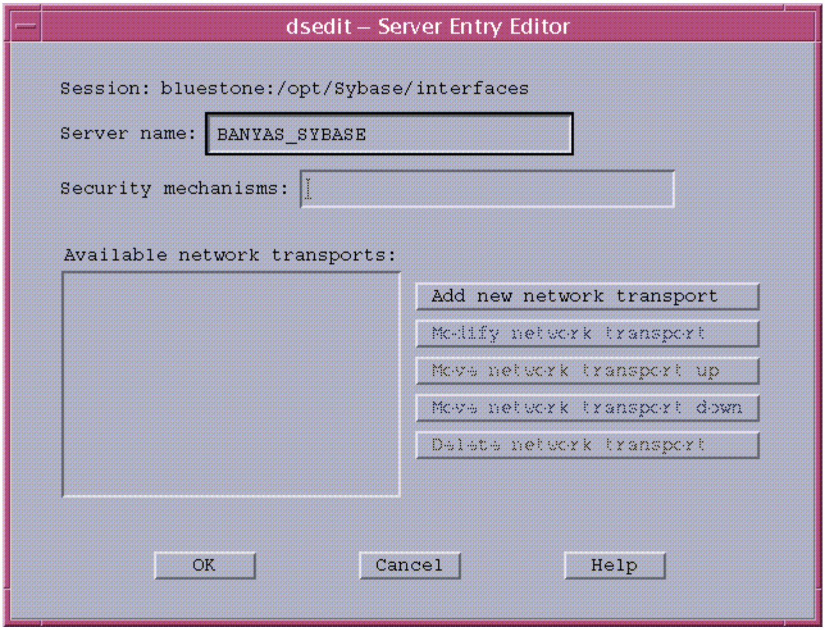
dsedit – Directory Service Session

Session: bluestone:/opt/Sybase/interfaces

Available servers:

BLUESTONE_SYBASE
bluestone_back

4. Highlight the Sybase server from the list of Available Servers and click the Add New Server Entry button. The Server Entry Editor dialog appears:



dsedit – Server Entry Editor

Session: bluestone:/opt/Sybase/interfaces

Server name: BANYAS_SYBASE

Security mechanisms:

Available network transports:

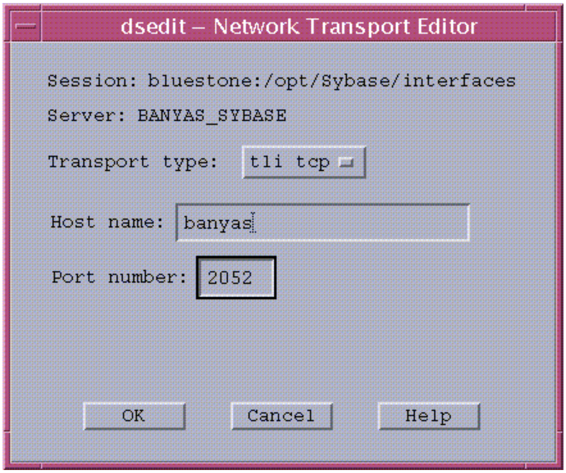
Add new network transport
Modify network transport
Move network transport up
Move network transport down
Delete network transport
OK
Cancel
Help

5. Type in the name of the secondary Sybase server (not to be confused with the server’s host name) and click on the Add New Network Transport button.

Note: The Sybase server name on a UNIX platform is typically *hostname*_SYBASE. The Sybase server name on an NT platform is typically the same as the host name.

In this example, we are adding a UNIX server named BANYAS_SYBASE to the Sybase Interfaces file.

A secondary dialog entitled Network Transport Editor appears:



dsedit – Network Transport Editor

Session: bluestone:/opt/Sybase/interfaces

Server: BANYAS_SYBASE

Transport type: tli tcp

Host name: banyas

Port number: 2052

6. Select tli tcp as the Transport Type, enter the secondary server's host name (BANYAS, in this example), enter the port number that is configured for Sybase connections on the secondary server, and click OK.

Note: The port number for TREND servers installed on a UNIX system is typically 2052. The port number for TREND servers installed on a Windows NT system is typically 5000.

This returns you to the Server Entry Editor dialog.

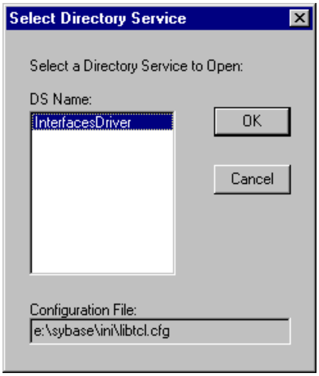
7. Click OK. This returns you to the Directory Service Session dialog.
8. Click the Close Session button. This returns you to the Select a Directory Service dialog.
9. Click Exit.
10. Click the Yes button when your are prompted about exiting dsedit. This returns you to the UNIX command line.

You can validate the result by viewing the $SYBASE/interfaces file.

Creating the Sybase Interfaces File on Windows Systems

Use the Dsedit utility to install a secondary server interfaces file entry for Sybase 11.5.1 on Windows systems. Perform these steps:

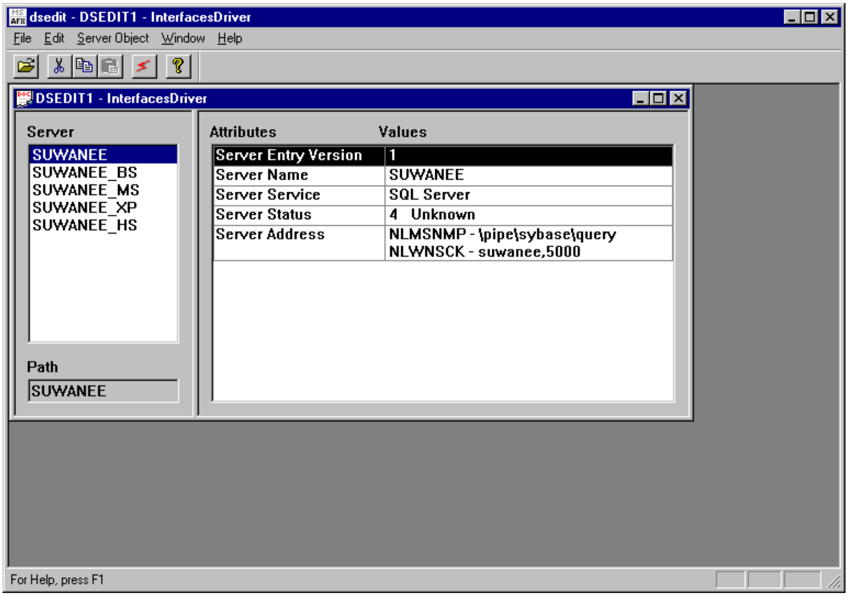
1. As user trendadm, select Programs>Sybase>Dsedit from the Start menu. The Dsedit window appears with the Select Directory Service window displayed within it.



InterfacesDriver

e:\sybase\ini\libtcl.cfg

2. Select InterfacesDriver and click OK. The DSEDIT1 - InterfacesDriver window appears.


dsedit - DSEDIT1 - InterfacesDriver
File
Edit
Server Object
Window
Help
DSEDIT1 - InterfacesDriver
For Help, press F1

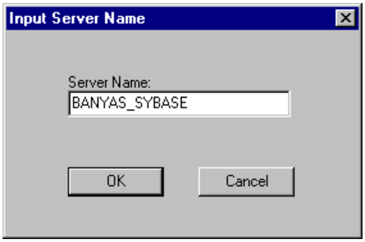
Server

SUWANEE
SUWANEE_BS
SUWANEE_MS
SUWANEE_XP
SUWANEE_HS

Path: SUWANEE

Attributes	Values
Server Entry Version	1
Server Name	SUWANEE
Server Service	SQL Server
Server Status	4 Unknown
Server Address	NLMSNMP - \pipe\sybase\query NLWNSCK - suwanee,5000

3. From the Server Object pull-down menu, select Add. The Input Server Name window appears.

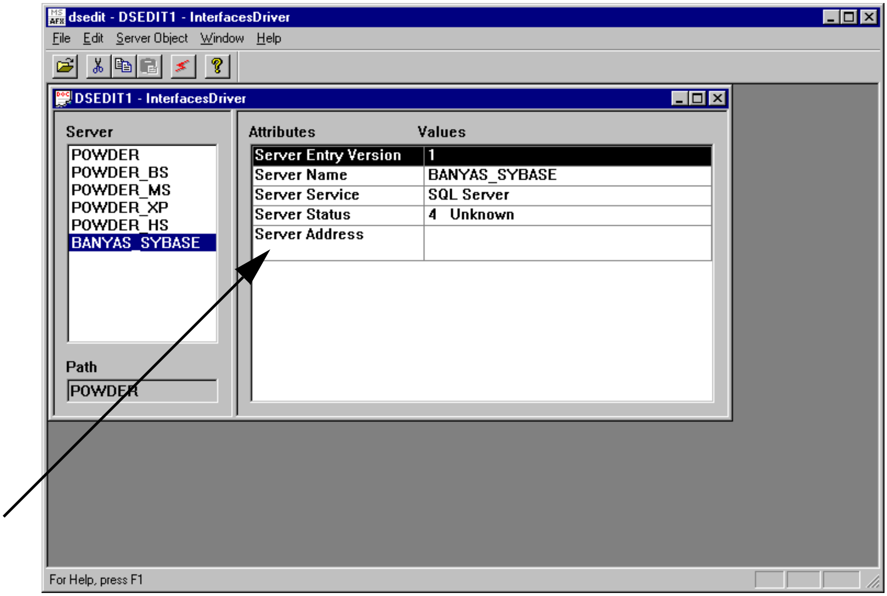


Server Name: BANYAS_SYBASE

4. Enter the database server name (In this example, the server name is BANYAS_SYBASE, which is on a UNIX platform.)

Note: The Sybase server name on a UNIX platform is typically *hostname*_SYBASE. The Sybase server name on an NT platform is typically the same as the host name.

5. Click OK. The DSEDIT1 - InterfacesDriver window reappears.

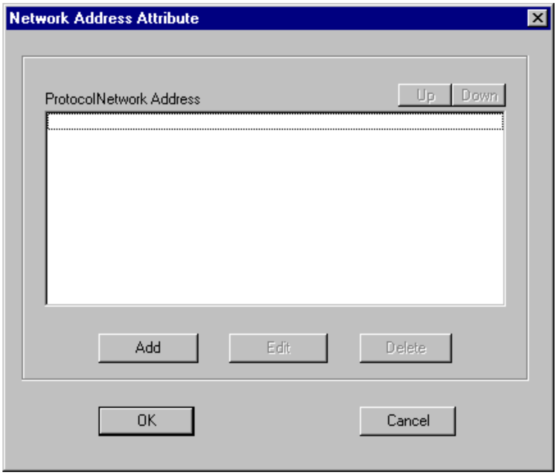

dsedit - DSEDIT1 - InterfacesDriver
File Edit Server Object Window Help
DSEDIT1 - InterfacesDriver
Server
Path
For Help, press F1

POWDER
POWDER_BS
POWDER_MS
POWDER_XP
POWDER_HS
BANYAS_SYBASE

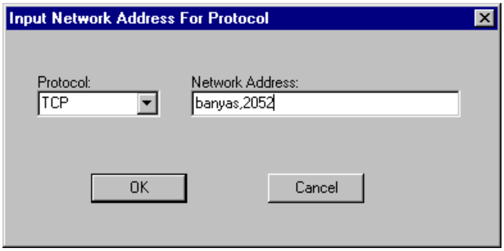
Attributes	Values
Server Entry Version	1
Server Name	BANYAS_SYBASE
Server Service	SQL Server
Server Status	4 Unknown
Server Address	

Path: POWDER

6. Double-click on the Server Address attribute in the Attributes column. The Network Address Attribute window appears.



7. Click Add. The Input Network Address For Protocol window appears.



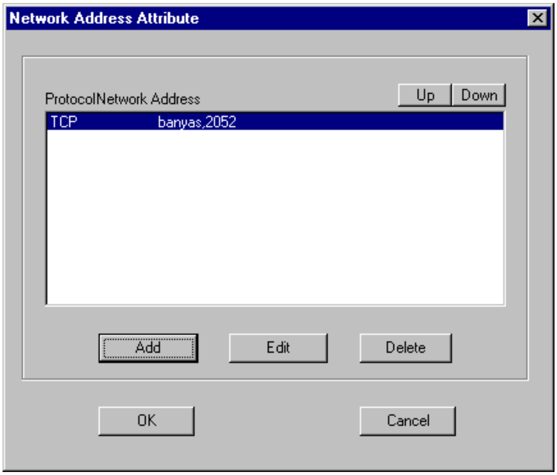
Protocol: TCP

Network Address: banyas,2052

8. In the Protocol list box, click the downarrow and choose TCP. In the Network Address field, type the host name of the server (banyas, in this example), a comma, and the port number to which the server listens (2052, in this example).

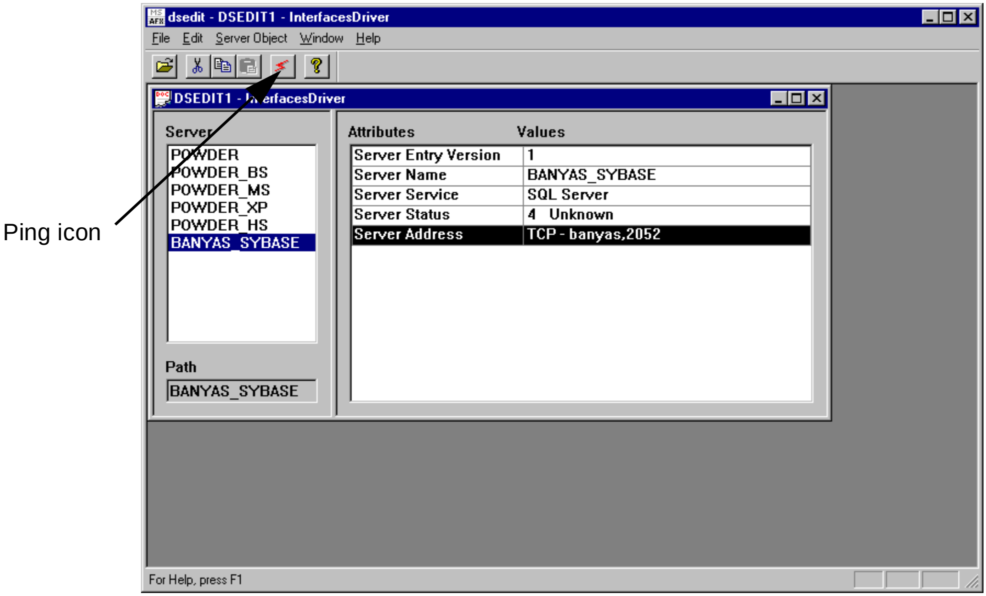
Note: The port number for TREND servers installed on a UNIX system is typically 2052. The port number for TREND servers installed on a Windows NT system is typically 5000.

9. Click OK. The Network Address Attribute window reappears with your entry.



Protocol	Network Address
TCP	banyas,2052

10. Click OK. The DSEDIT1 - InterfacesDriver window reappears.

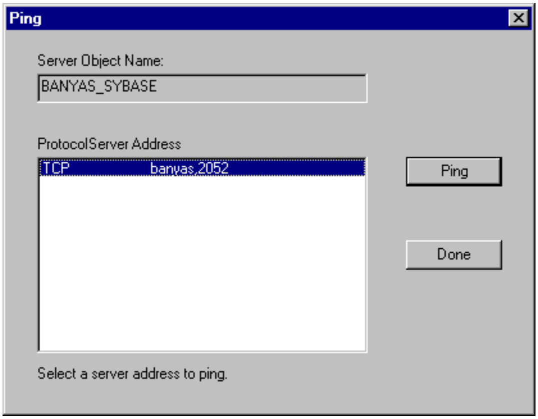

dsedit - DSEDIT1 - InterfacesDriver
File
Edit
Server Object
Window
Help
DSEDIT1 - InterfacesDriver
Ping icon
For Help, press F1

Server
POWDER
POWDER_BS
POWDER_MS
POWDER_XP
POWDER_HS
BANYAS_SYBASE

Path
BANYAS_SYBASE

Attributes	Values
Server Entry Version	1
Server Name	BANYAS_SYBASE
Server Service	SQL Server
Server Status	4 Unknown
Server Address	TCP - banyas,2052

11. At this point, test the connection to the specified server by clicking on the Ping icon (the red lightning bolt icon below the menu bar). The Ping window appears.

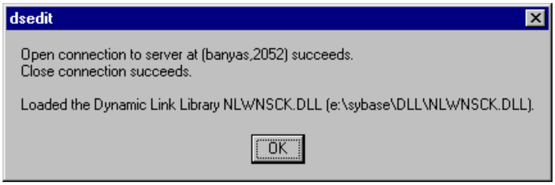


Server Object Name: BANYAS_SYBASE

TCP banyas,2052

Select a server address to ping.

12. Click the Ping button. A message similar to the following appears in the dsedit window if you have successfully opened a connection to the specified server.



Open connection to server at (banyas,2052) succeeds.
Close connection succeeds.

Loaded the Dynamic Link Library NLWNSCK.DLL (e:\sybase\DLL\NLWNSCK.DLL).

Note: If the message you receive indicates that the server is not recognized, you have incorrectly specified the host name of the server or the port number or the server is down. In this case, verify the host name of the server and port number. If you are still having difficulties, contact DeskTalk Systems Technical Support.

13. Click OK to return to the Ping window. In the Ping window, click Done to return to the DSEDIT1 - InterfacesDriver window. Select Exit from the File menu to exit from the Dsedit utility.

E Discovery Considerations

This appendix describes how TREND stores nodes and how you can modify Discovery files. The discussion addresses the advanced user who needs an understanding of the schema and wants to expand the list of devices that TREND auto-discovers. A Wellfleet device is used as an example in this discussion.

TREND uses two node lists:

- A list of discovered nodes stored in the dsi_node_list table.
- A list of manually entered nodes stored in the dsi_nodes table.

The user interface allows you to look at node types to see the discovered nodes that are in each type, but does not allow you to see just a list of all discovered nodes.

However, you can use the following commands from an isql session to see just the list of all discovered nodes:

```
isql -Udsi_dpipe -Pdsi_dpipe
1>select dsi_target_name from dsi_node_list
2>go
```

The following query lists all devices that have been discovered but not added to a type:

```
isql -Udsi_dpipe -Pdsi_dpipe
1>select dsi_target_name from dsi_node_list
2>where dsi_target_name not in (select name_ from dl_type)
3>go
```

This list identifies:

- The routers or other devices were IP-discovered but not type-discovered.
- The routers or other devices that were not IP-discovered or type-discovered.

In actuality, there are two discovery modes:

IP discovery The dsi_status value (on the dsi_nodes_list table) for a device is incremented each time the device fails to respond. If the device responds, its dsi_status value is reset to 0.

If the dsi_status for a device exceeds either the default of 10 or the value specified with the -a option, the device is deleted from dsi_nodes, dsi_snmp_nodes, dsi_nodes_list, and dl_type.

Type discovery Type discovery is configured to be a scheduled activity each morning if you install a report package via AutoPilot. Type discovery uses only dsi_node entries and tries to identify the type that applies to the node. If the node is already member of a type being tested, it is skipped and no determination of whether it should still belong is made.

For example, consider the situation where the Router_Interfaces type is not displayed in the Define Types dialog because no nodes have been type-cast yet. Having no nodes matched against the type definition causes the following messages to be written to trend.log:

```
mw_collect: Thu Jan 22 16:45:00 1998 - Skip type
Router_Interfaces,not found in DL_TYPES & DSI_NODES
```

The discover did not work because the value returned via the sysObjectID did not match the value in the .dis files (in the $DPIPE_HOME\scripts directory), which are designed to find major-brand-vendor routers. The sysObjectID is probably different for the devices being referenced.

DeskTalk Systems has placed additional files on its FTP server in ~ftp/pub/TREND/Discover_Files. Currently, the files are for Router_Interfaces and Baseline. As customers and Desktalk Systems engineers find other devices that are not being discovered, we will place the needed files first on the ftp server and then in TREND.

For example, the Router_Interfaces additions will match the OID string for Wellfleet, 1.3.6.1.4.1.18 and cause the devices matching this string to be placed in the Router_Interfaces type.

If this additional discovery file does not solve the issue, you can use the MIBwalker identify to help identify the setting necessary for recognizing the devices:

1. Bring up MIBwalker.
2. Click on the bold **system** branch to highlight it.
3. Select Change Node from the Tools menu.
4. Choose an address of the device and click OK.
5. Select SNMP Tool from the Tools menu and click Get.
6. In the data window, look for the line for sysObjectID.
7. Copy one of the existing discovery files, for example, the 3com.dis file (in the $DPIPE_HOME\scripts directory), to a new name such as new_device.dis (into the $DPIPE_HOME\scripts directory).
8. Edit the new file to remove the snippet in quotes on the right side of the equal sign and replace it with the sysObjectID result (from the MIBwalker Value column) shown after the **enterprises.** snippet. For example:

new_device.dis was:	`1.3.6.1.4.1.43.1.4.*`
MIBwalker shows:	`enterprises.141.1.1.6050`
new_device.dis becomes:	`1.3.6.1.4.1.141.1.1.6050`

This affects only the Router Interfaces package, but it provides a starting point to understanding how type discovery is processing, so that you can make necessary revisions as you identify new devices for TREND to monitor.

TREND

F Relocating the TREND Database

If you need to move the TREND database from one host to another, DeskTalk Systems recommends creating and using backup Sybase servers on both host A (the source host) and host B (the target host). Follow these steps:

1. Install a backup Sybase server on host A. Make sure both the Sybase server and the backup Sybase server are running on host A. See “Backup Server” on page B-15 for detailed instructions.
2. Install TREND on host B.

CAUTION *The database size on host B must be equal to or greater than the one on host A for these steps to be completed.*

3. On host B, turn off trendtimer so there is no automatic database activity.
4. Install a backup Sybase server on host B. Make sure both the Sybase server and the backup Sybase server are running on host B. See “Backup Server” on page B-15 for detailed instructions.
5. Turn off trendtimer on host A.
6. Wait for any collections to complete. Then, dump the database to an available disk file. Make sure you have plenty of space available to create the dump file. You can get latest backup by starting an isql session. For example:

```
isql -Usa -P
1>sp_addumpdevice 'disk',diskdump,'<path-to-diskspace>/
diskdump.dat'
2>go
1>dump database dpipe_db to diskdump with noinit
2>go
```

7. Mount the A <path-to-diskspace> file system for export on B

   or:

   ftp the 'diskdump.dat' file to host B depending on whether you have available disk resources on B to handle the temporary storage of the dumped database file.

8. Load the backup on host B by starting an isql session. For example:

```
isql -Usa -P
1>sp_addumpdevice 'disk',diskdump,'<new-path-to-
diskspace>/diskdump.dat'
2>go
1>load database dpipe_db from diskdump
2>go
1>online database dpipe_dp
2>go
```

9. After this completes, unmount the A <path-to-diskspace> file system on B or delete the 'diskdump.dat' file on host B, depending on the method you chose to get the file onto host B.

10. Once you have confirmed that the database on host B looks complete, you can turn on trendtimer on host B to see how things work.

Check the following:

1. Policies. You must change the value of the Poll From field to the new host name for each policy listed. Verify that your text change is correct by using it with a ping command.

2. Environment variables (for trendadm and all other users). Make sure that the files of $DPIPE_HOME/lib/Cshrc and $DPIPE_HOME/lib/Profile have been changed (either by new installation of by editing) to have new settings, specifically DISPLAY, DSQUERY, and DSI_ELMHOST.

3. Database revisions. You must change the dsquery column in the tl_groups table (for all entries) to be the new Sybase environment variable value for the host B system. You can do this with ISQL using the following commands:

   ```
   isql -Udsi_dpipe -Pdsi_dpipe
   1> update tl_groups set dsquery = <Sybase-DSQUERY-value>,
   2> data_db = <Sybase-DSQUERY-value>
   3> go
   ```

4. Reporting. If you have installed packages with AutoPilot or have added reports, then copy the following files and directories from host A to host B:

   - $DPIPE_HOME/lib/print.* files
   - $DPIPE_HOME/lib/*.info files
   - $DPIPE_HOME/reports directory
   - $DPIPE_HOME/scripts directory

   1. Make sure that any custom report files (.qss/.qgr/.gos), which can reside in any directory on host A, are copied to host B.
   2. Make sure that any custom scripts, which can reside in any directory on host A, are copied to host B.
   3. Edit the trendtimer.sched file on host B to match the times and entries of the trendtimer.sched file on host A, especially the lines regarding trend_proc -f xxx.pro for the packages and the lines regarding any custom reports/scripts.

5. Trend license. Permanent keys are tied to hostid and ip-address. Therefore, a new license may be required. Supply the new details to DeskTalk Systems so that we can generate a new key. (In the short-term, DeskTalk Systems can provide a temporary key.)

TREND

TREND

Index

TREND